Vorlesungsskript

Theoretische Informatik 2

Wintersemester 2009/10

Prof. Dr. Johannes Köbler
Humboldt-Universität zu Berlin
Lehrstuhl Komplexität und Kryptografie

19. November 2009

Inhaltsverzeichnis

Inhaltsverzeichnis

1 Einleitung

In der Vorlesung ThI 1 standen die mathematischen Grundlagen der Informatik im Vordergrund. Insbesondere lernten Sie, wie man folgerichtig argumentiert und wie man formale Beweise führt. Als universelle Sprache der Mathematik lernten Sie dabei die mathematische Logik kennen, insbesondere die Aussagenlogik und darauf aufbauend die Prädikatenlogik. In dieser Sprache lassen sich nicht nur algebraische und relationale Strukturen modellieren, sondern auch Rechenmaschinen wie zum Beispiel die Turingmaschine.

Ein weiteres wichtiges Thema der Vorlesung ThI1 war die Frage, welche Probleme algorithmisch lösbar sind.

Themen der Vorlesung ThI1

- Mathematische Grundlagen der Informatik, Beweise führen, Modellierung (Aussagenlogik, Prädikatenlogik)
- Welche Probleme sind lösbar? (Berechenbarkeitstheorie)

Dagegen stehen in dieser Vorlesung folgende Fragen im Mittelpunkt.

Themen der Vorlesung ThI2

- Welche Rechenmodelle sind für bestimmte Aufgaben adäquat? (Automatentheorie)
- Welcher Aufwand ist zur Lösung eines algorithmischen Problems nötig? (Komplexitätstheorie)

Schließlich wird es in der Vorlesung ThI 3 in erster Linie um folgende Frage gehen.

Thema der Vorlesung ThI3

- Wie lassen sich eine Reihe von praktisch relevanten Problemstellungen möglichst effizient lösen? (Algorithmik)

Rechenmaschinen spielen in der Informatik eine zentrale Rolle. Hier beschäftigen wir uns mit mathematischen Modellen für Maschinentypen von unterschiedlicher Berechnungskraft. In der Vorlesung Theoretische Informatik 1 wurde die Turingmaschine als ein universales Berechnungsmodell eingeführt. In ThI3 wird das etwas flexibere Modell der Registermaschine (engl. random access machine; RAM) benutzt. Dieses Modell erlaubt den unmittelbaren Lese- und Schreibzugriff (random access) auf eine beliebige Speichereinheit (Register). Hier betrachten wir Einschränkungen des TM-Modells, die vielfältige praktische Anwendungen haben, wie z.B. endliche Automaten (DFA, NFA), Kellerautomaten (PDA, DPDA) etc.

Der Begriff *Algorithmus* geht auf den persischen Gelehrten Muhammed Al Chwarizmi (8./9. Jhd.) zurück. Der älteste bekannte nicht-triviale Algorithmus ist der nach *Euklid* benannte Algorithmus zur Berechnung des größten gemeinsamen Teilers zweier natürlicher Zahlen (300 v. Chr.). Von einem Algorithmus wird erwartet, dass er jede *Problemeingabe* nach endlich vielen Rechenschritten löst (etwa durch Produktion einer Ausgabe). Ein Algorithmus ist ein „Verfahren“ zur Lösung eines Berechnungsproblems, das sich prinzipiell auf einer Turingmaschine implementieren lässt (Church-Turing-These).

Wir betrachten zunächst nur Entscheidungsprobleme, was der Berechnung von $\{0,1\}$-wertigen Funktionen entspricht. Problemeingaben können Zahlen, Formeln, Graphen etc. sein. Diese werden über einem *Eingabealphabet* Σ kodiert.

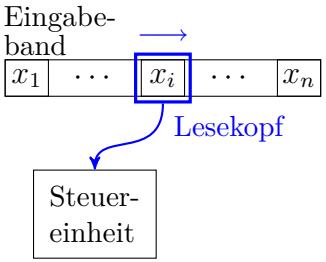
Definition 1. *Ein **Alphabet** ist eine geordnete endliche Menge $\Sigma = \{a_1, \ldots, a_m\}$, $m \geq 1$, von **Zeichen**. Eine Folge $x = x_1 \ldots x_n \in \Sigma^n$ heißt **Wort** (der **Länge** n). Die Menge aller Wörter über Σ ist*

$$\Sigma^* = \bigcup_{n \geq 0} \Sigma^n = \{x_1 \cdots x_n \mid n \geq 0 \text{ und } x_i \in \Sigma \text{ für } i = 1, \ldots, n\}.$$

*Das (einzige) Wort der Länge $n = 0$ ist das **leere Wort**, welches wir mit ε bezeichnen. Jede Teilmenge $L \subseteq \Sigma^*$ heißt **Sprache** über dem Alphabet Σ.*

2 Reguläre Sprachen

2.1 Endliche Automaten



Ein endlicher Automat ist eine „abgespeckte“ Turingmaschine, die nur kostant viel Speicherplatz benötigt und bei Eingaben der Länge n nur n Rechenschritte ausführt. Um die gesamte Eingabe lesen zu können, muss der Automat also in jedem Schritt ein Zeichen der Eingabe verarbeiten.

Definition 2. *Ein **endlicher Automat** (kurz: DFA; deterministic finite automaton) wird durch ein 5-Tupel $M = (Z, \Sigma, \delta, q_0, E)$ beschrieben, wobei*

- *$Z \neq \emptyset$ eine endliche Menge von **Zuständen**,*
- *Σ das **Eingabealphabet**,*
- *$\delta : Z \times \Sigma \to Z$ die **Überführungsfunktion**,*
- *$q_0 \in Z$ der **Startzustand** und*
- *$E \subseteq Z$ die Menge der **Endzustände** ist.*

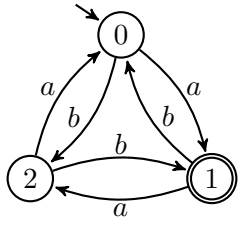
*Die von M **akzeptierte** oder **erkannte Sprache** ist*

$$L(M) = \left\{ x_1 \cdots x_n \in \Sigma^* \,\middle|\, \begin{array}{c} \exists\, q_1, \ldots, q_{n-1} \in Z, q_n \in E: \\ \delta(q_i, x_{i+1}) = q_{i+1} \text{ für } i = 0, \ldots, n-1 \end{array} \right\}.$$

Beispiel 3. *Betrachte den DFA $M = (Z, \Sigma, \delta, 0, E)$ mit $Z = \{0, 1, 2\}$, $\Sigma = \{a, b\}$, $E = \{1\}$ und der Überführungsfunktion*

δ	0	1	2
a	1	2	0
b	2	0	1

Graphische Darstellung:



Der Startzustand wird meist durch einen Pfeil und Endzustände werden durch einen doppelten Kreis gekennzeichnet. ◁

Bezeichne $\hat{\delta}(q, x)$ denjenigen Zustand, in dem sich M nach Lesen von x befindet, wenn M im Zustand q gestartet wird. Dann können wir die Funktion

$$\hat{\delta} : Z \times \Sigma^* \to Z$$

induktiv wie folgt definieren. Für $q \in Z$, $x \in \Sigma^*$ und $a \in \Sigma$ sei

$$\begin{aligned} \hat{\delta}(q, \varepsilon) &= q, \\ \hat{\delta}(q, xa) &= \delta(\hat{\delta}(q, x), a). \end{aligned}$$

Die von M erkannte Sprache lässt sich nun auch in der Form

$$L(M) = \{x \in \Sigma^* \mid \hat{\delta}(q_0, x) \in E\}$$

schreiben.

Behauptung 1. *Der DFA M aus Beispiel 3 akzeptiert die Sprache*

$$L(M) = \{x \in \Sigma^* \mid \#_a(x) - \#_b(x) \equiv 1 \pmod 3\},$$

wobei $\#_a(x)$ die Anzahl der Vorkommen des Buchstabens a in x bezeichnet und $j \equiv k \pmod m$ bedeutet, dass $j - k$ durch m teilbar ist. Für $j \equiv k \pmod m$ schreiben wir im Folgenden auch kurz $j \equiv_m k$.

Beweis. Da M nur den Endzustand 1 hat, ist $L(M) = \{x \in \Sigma^* \mid \hat{\delta}(0,x) = 1\}$. Daher reicht es, folgende Kongruenzgleichung zu zeigen:

$$\hat{\delta}(0,x) \equiv_3 \#_a(x) - \#_b(x).$$

Wir beweisen die Kongruenz induktiv über die Länge n von x.

Induktionsanfang ($n = 0$): klar, da $\hat{\delta}(0,\varepsilon) = \#_a(\varepsilon) = \#_b(\varepsilon) = 0$ ist.

Induktionsschritt ($n \rightsquigarrow n+1$): Sei $x = x_1 \cdots x_{n+1}$ gegeben und sei $i = \hat{\delta}(0, x_1 \cdots x_n)$. Nach IV ist

$$i \equiv_3 \#_a(x_1 \cdots x_n) - \#_b(x_1 \cdots x_n).$$

Wegen $\delta(i,a) \equiv_3 i+1$ und $\delta(i,b) \equiv_3 i-1$ folgt

$$\delta(i, x_{n+1}) \equiv_3 i + \#_a(x_{n+1}) - \#_b(x_{n+1}) = \#_a(x) - \#_b(x).$$

Folglich ist

$$\hat{\delta}(0,x) = \delta(\hat{\delta}(0, x_1 \cdots x_n), x_{n+1}) = \delta(i, x_{n+1}) \equiv_3 \#_a(x) - \#_b(x).$$

■

Eine von einem DFA akzeptierte Sprache wird als **regulär** bezeichnet. Die zugehörige Sprachklasse ist

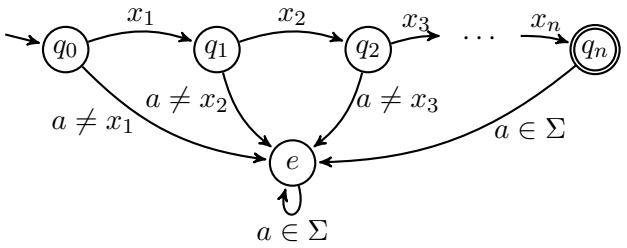
$$\mathsf{REG} = \{L(M) \mid M \text{ ist ein DFA}\}.$$

Um ein intuitives Verständnis für die Berechnungskraft von DFAs zu entwickeln, werden wir Antworten auf folgende Frage suchen.

Frage: Welche Sprachen gehören zu REG und welche nicht?

Dabei legen wir unseren Überlegungen ein beliebiges aber fest gewähltes Alphabet $\Sigma = \{a_1, \ldots, a_m\}$ zugrunde.

Beobachtung 4. *Alle Sprachen, die aus einem einzigen Wort $x = x_1 \cdots x_n \in \Sigma^*$ bestehen (diese Sprachen werden auch als* Singletonsprachen *bezeichnet), sind regulär. Für folgenden DFA M gilt nämlich $L(M) = \{x\}$.*



Formal lässt sich M also durch das Tupel $M = (Z, \Sigma, \delta, q_0, E)$ mit $Z = \{q_0, \ldots, q_n, e\}$, $E = \{q_n\}$ und der Überführungsfunktion

$$\delta(q, a_j) = \begin{cases} q_{i+1}, & q = q_i \text{ für ein } i \text{ mit } 0 \leq i \leq n-1 \text{ und } a_j = x_{i+1} \\ e, & \text{sonst} \end{cases}$$

beschreiben.

Als nächstes betrachten wir Abschlusseigenschaften der Sprachklasse REG.

Definition 5. *Ein* **(k-stelliger) Sprachoperator** *ist eine Abbildung op, die k Sprachen $L_1, \ldots, L_k$ auf eine Sprache $op(L_1, \ldots, L_k)$ abbildet.*

Beispiel 6. *Der 2-stellige Schnittoperator bildet zwei Sprachen L_1 und L_2 auf die Sprache $L_1 \cap L_2$ ab.* ◁

Definition 7. *Eine Sprachklasse $\mathcal{K}$ heißt unter op* **abgeschlossen**, *wenn gilt:*

$$L_1, \ldots, L_k \in \mathcal{K} \Rightarrow op(L_1, \ldots, L_k) \in \mathcal{K}.$$

Der **Abschluss** *von $\mathcal{K}$ unter op ist die kleinste Sprachklasse $\mathcal{K}'$, die $\mathcal{K}$ enthält und unter op abgeschlossen ist.*

Definition 8. *Für eine Sprachklasse $\mathcal{C}$ bezeichne co-$\mathcal{C}$ die Klasse $\{\bar{L} \mid L \in \mathcal{C}\}$ aller Komplemente von Sprachen in $\mathcal{C}$.*

Es ist leicht zu sehen, dass $\mathcal{C}$ genau dann unter Komplementbildung abgeschlossen ist, wenn *co-*$\mathcal{C} = \mathcal{C}$ ist.

Beobachtung 9. *Mit $L_1, L_2 \in$ REG sind auch die Sprachen $\overline{L_1} = \Sigma^* \setminus L_1$, $L_1 \cap L_2$ und $L_1 \cup L_2$ regulär. Sind nämlich $M_i = (Z_i, \Sigma, \delta_i, q_0, E_i)$, $i = 1, 2$, DFAs mit $L(M_i) = L_i$, so akzeptiert der DFA*

$$\overline{M_1} = (Z_1, \Sigma, \delta_1, q_0, Z_1 \setminus E_1)$$

das Komplement $\overline{L_1}$ von L_1. Der Schnitt $L_1 \cap L_2$ von L_1 und L_2 wird dagegen von dem DFA

$$M = (Z_1 \times Z_2, \Sigma, \delta, (q_0, q_0), E_1 \times E_2)$$

mit

$$\delta((q, p), a) = (\delta_1(q, a), \delta_2(p, a))$$

akzeptiert (M wird auch **Kreuzproduktautomat** *genannt). Wegen $L_1 \cup L_2 = \overline{(\overline{L_1} \cap \overline{L_2})}$ ist dann aber auch die Vereinigung von L_1 und L_2 regulär. (Wie sieht der zugehörige DFA aus?)*

Aus Beobachtung 9 folgt, dass alle endlichen und alle co-endlichen Sprachen regulär sind. Da die in Beispiel 3 betrachtete Sprache weder endlich noch co-endlich ist, haben wir damit allerdings noch nicht alle regulären Sprachen erfasst.

Es stellt sich die Frage, ob REG neben den mengentheoretischen Operationen Schnitt, Vereinigung und Komplement unter weiteren Operationen wie etwa der **Produktbildung**

$$L_1 L_2 = \{xy \mid x \in L_1, y \in L_2\}$$

(auch **Verkettung** oder **Konkatenation** genannt) oder der Bildung der **Sternhülle**

$$L^* = \bigcup_{n \geq 0} L^n$$

abgeschlossen ist. Die n-fache Potenz L^n von L ist dabei induktiv definiert durch

$$L^0 = \{\varepsilon\},\ L^{n+1} = L^n L.$$

Die **Plushülle** von L ist

$$L^+ = \bigcup_{n \geq 1} L^n = LL^*.$$

Ist $L_1 = \{x\}$ eine Singletonsprache, so schreiben wir für das Produkt $\{x\}L_2$ auch einfach xL_2.

Im übernächsten Abschnitt werden wir sehen, dass die Klasse REG als der Abschluss der endlichen Sprachen unter Vereinigung, Produktbildung und Sternhülle charakterisierbar ist.

Beim Versuch, einen endlichen Automaten für das Produkt $L_1 L_2$ zweier regulärer Sprachen zu konstruieren, stößt man auf die Schwierigkeit, den richtigen Zeitpunkt für den Übergang von (der Simulation von) M_1 zu M_2 zu finden. Unter Verwendung eines nichtdeterministischen Automaten lässt sich dieses Problem jedoch leicht beheben, da dieser den richtigen Zeitpunkt „erraten“ kann.

Im nächsten Abschnitt werden wir nachweisen, dass auch nichtdeterministische endliche Automaten nur reguläre Sprachen erkennen können.

2.2 Nichtdeterministische endliche Automaten

Definition 10. *Ein* ***nichtdeterministischer endlicher Automat*** *(kurz: NFA; nondeterministic finite automaton) $N = (Z, \Sigma, \delta, Q_0, E)$ ist ähnlich aufgebaut wie ein DFA, nur dass er mehrere Startzustände (zusammengefasst in der Menge $Q_0 \subseteq Z$) haben kann und seine Überführungsfunktion die Form*

$$\delta : Z \times \Sigma \to \mathcal{P}(Z)$$

hat. Hierbei bezeichnet $\mathcal{P}(Z)$ *die* Potenzmenge *(also die Menge aller Teilmengen) von* Z*. Diese wird auch oft mit* 2^Z *bezeichnet. Die von* N *akzeptierte Sprache ist*

$$L(N) = \left\{ x_1 \cdots x_n \in \Sigma^* \;\middle|\; \begin{array}{c} \exists\, q_0 \in Q_0, q_1, \ldots, q_{n-1} \in Z, q_n \in E\colon \\ q_{i+1} \in \delta(q_i, x_{i+1}) \text{ für } i = 0, \ldots, n-1 \end{array} \right\}.$$

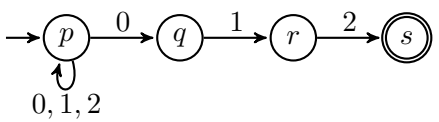
Ein NFA kann also nicht nur eine, sondern mehrere verschiedene Rechnungen ausführen. Die Eingabe gehört bereits dann zu $L(N)$, wenn bei einer dieser Rechnungen nach Lesen des gesamten Eingabewortes ein Endzustand erreicht wird.

Im Gegensatz zu einem DFA, dessen Überführungsfunktion auf der gesamten Menge $Z \times \Sigma$ definiert ist, kann ein NFA „stecken bleiben“. Das ist dann der Fall, wenn er in einen Zustand q gelangt, in dem das nächste Eingabezeichen x_i wegen $\delta(q, x_i) = \emptyset$ nicht gelesen werden kann.

Beispiel 11. *Betrachte den NFA* $N = (Z, \Sigma, \delta, Q_0, E)$ *mit Zustandsmenge* $Z = \{p, q, r, s\}$*, Eingabealphabet* $\Sigma = \{0, 1, 2\}$*, Start- und Endzustandsmenge* $Q_0 = \{p\}$ *und* $E = \{s\}$ *sowie der Überführungsfunktion*

δ	p	q	r	s
0	$\{p, q\}$	$\emptyset$	$\emptyset$	$\emptyset$
1	$\{p\}$	$\{r\}$	$\emptyset$	$\emptyset$
2	$\{p\}$	$\emptyset$	$\{s\}$	$\emptyset$

Graphische Darstellung:



Offensichtlich akzeptiert N *die Sprache* $L(N) = \{x012 \mid x \in \Sigma^*\}$ *aller Wörter, die mit dem Suffix* 012 *enden.* ◁

Beobachtung 12. *Sind* $N_i = (Z_i, \Sigma, \delta_i, Q_i, E_i)$ *(*$i = 1, 2$*) NFAs, so werden auch die Sprachen* $L(N_1)L(N_2)$ *und* $L(N_1)^*$ *von einem NFA erkannt. Wir können* $Z_1 \cap Z_2 = \emptyset$ *annehmen. Dann akzeptiert der NFA*

$$N = (Z_1 \cup Z_2, \Sigma, \delta, Q_1, E)$$

mit

$$\delta(p, a) = \begin{cases} \delta_1(p, a), & p \in Z_1 \setminus E_1, \\ \delta_1(p, a) \cup \bigcup_{q \in Q_2} \delta_2(q, a), & p \in E_1, \\ \delta_2(p, a), & \text{sonst} \end{cases}$$

und

$$E = \begin{cases} E_1 \cup E_2, & Q_2 \cap E_2 \neq \emptyset \\ E_2, & \text{sonst} \end{cases}$$

die Sprache $L(N_1)L(N_2)$ *und der NFA*

$$N^* = (Z_1 \cup \{q_{neu}\}, \Sigma, \delta^*, Q_1 \cup \{q_{neu}\}, E_1 \cup \{q_{neu}\})$$

mit

$$\delta^*(p, a) = \begin{cases} \delta(p, a) \cup \bigcup_{q \in Q_1} \delta(q, a), & p \in E_1, \\ \delta(p, a), & \text{sonst} \end{cases}$$

die Sprache $L(N_1)^*$*.*

Satz 13 (Rabin und Scott)**.**
$\mathsf{REG} = \{L(N) \mid N \text{ ist ein NFA}\}$.

Beweis. Die Inklusion von links nach rechts ist klar, da jeder DFA auch als NFA aufgefasst werden kann. Für die Gegenrichtung konstruieren wir zu einem NFA $N = (Z, \Sigma, \delta, Q_0, E)$ einen DFA $M = (\mathcal{P}(Z), \Sigma, \delta', Q_0, E')$ mit $L(M) = L(N)$. Wir definieren die Überführungsfunktion $\delta' : \mathcal{P}(Z) \times \Sigma \to \mathcal{P}(Z)$ von M mittels

$$\delta'(Q, a) = \bigcup_{q \in Q} \delta(q, a).$$

Die Menge $\delta'(Q, a)$ enthält also alle Zustände, in die N gelangen kann, wenn N ausgehend von einem beliebigen Zustand $q \in Q$ das Zeichen a liest. Intuitiv bedeutet dies, dass der DFA M den NFA N simuliert, indem M in seinem aktuellen Zustand Q die Information speichert, in welchen Zuständen sich N momentan befinden könnte. Für die Erweiterung $\hat{\delta}' : \mathcal{P}(Z) \times \Sigma^* \to \mathcal{P}(Z)$ von δ' (siehe Seite 2) können wir nun folgende Behauptung zeigen:

$\hat{\delta}'(Q_0, x)$ enthält alle Zustände, die N ausgehend von einem Startzustand nach Lesen der Eingabe x erreichen kann.

Wir beweisen die Behauptung induktiv über die Länge n von x.

Induktionsanfang ($n = 0$): klar, da $\hat{\delta}'(Q_0, \varepsilon) = Q_0$ ist.

Induktionsschritt ($n - 1 \rightsquigarrow n$): Sei $x = x_1 \cdots x_n$ gegeben. Nach Induktionsvoraussetzung enthält

$$Q_{n-1} = \hat{\delta}'(Q_0, x_1 \cdots x_{n-1})$$

alle Zustände, die $N(x)$ in genau $n-1$ Schritten erreichen kann. Wegen

$$\hat{\delta}'(Q_0, x) = \delta'(Q_{n-1}, x_n) = \bigcup_{q \in Q_{n-1}} \delta(q, x_n)$$

enthält dann aber $\hat{\delta}'(Q_0, x)$ alle Zustände, die $N(x)$ in genau n Schritten erreichen kann.

Deklarieren wir nun diejenigen Teilmengen $Q \subseteq Z$, die mindestens einen Endzustand von N enthalten, als Endzustände des **Potenzmengenautomaten** M, d.h.

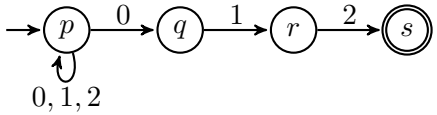
$$E' = \{Q \subseteq Z \mid Q \cap E \neq \emptyset\},$$

so folgt für alle Wörter $x \in \Sigma^*$:

$$\begin{aligned} x \in L(N) &\Leftrightarrow N(x) \text{ kann in genau } |x| \text{ Schritten einen Endzustand erreichen} \\ &\Leftrightarrow \hat{\delta}'(Q_0, x) \cap E \neq \emptyset \\ &\Leftrightarrow \hat{\delta}'(Q_0, x) \in E' \\ &\Leftrightarrow x \in L(M). \end{aligned}$$

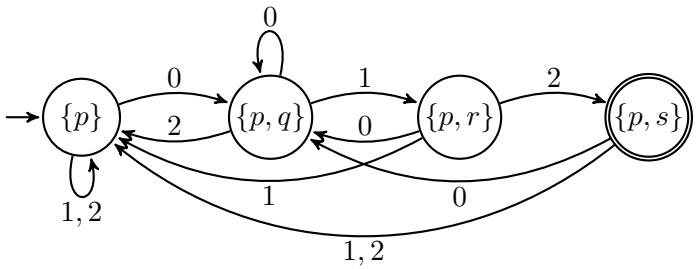
■

Beispiel 14. *Für den NFA $N = (Z, \Sigma, \delta, Q_0, E)$ aus Beispiel 11*



ergibt die Konstruktion des vorigen Satzes den folgenden DFA M (nach Entfernen aller vom Startzustand $Q_0 = \{p\}$ aus nicht erreichbaren Zustände):

δ'	0	1	2
$Q_0 = \{p\}$	$\{p, q\}$	$\{p\}$	$\{p\}$
$Q_1 = \{p, q\}$	$\{p, q\}$	$\{p, r\}$	$\{p\}$
$Q_2 = \{p, r\}$	$\{p, q\}$	$\{p\}$	$\{p, s\}$
$Q_3 = \{p, s\}$	$\{p, q\}$	$\{p\}$	$\{p\}$



◁

Im obigen Beispiel wurden für die Konstruktion des DFA M aus dem NFA N nur 4 der insgesamt $2^{\|Z\|} = 16$ Zustände benötigt, da die übrigen 12 Zustände in $\mathcal{P}(Z)$ nicht vom Startzustand $Q_0 = \{p\}$ aus erreichbar sind. Es gibt jedoch Beispiele, bei denen alle $2^{\|Z\|}$ Zustände in $\mathcal{P}(Z)$ für die Konstruktion des Potenzmengenautomaten benötigt werden (siehe Übungen).

Korollar 15. *Die Klasse* REG *der regulären Sprachen ist unter folgenden Operationen abgeschlossen:*

- *Komplement,*
- *Durchschnitt,*
- *Vereinigung,*
- *Produkt,*
- *Sternhülle.*

2.3 Reguläre Ausdrücke

Wir haben uns im letzten Abschnitt davon überzeugt, dass auch NFAs nur reguläre Sprachen erkennen können:

$$\mathsf{REG} = \{L(M) \mid M \text{ ist ein DFA}\} = \{L(N) \mid N \text{ ist ein NFA}\}.$$

In diesem Abschnitt werden wir eine weitere Charakterisierung der regulären Sprachen kennen lernen:

> REG ist die Klasse aller Sprachen, die sich mittels der Operationen Vereinigung, Durchschnitt, Komplement, Produkt und Sternhülle aus der leeren Menge und den Singletonsprachen bilden lassen.

Tatsächlich kann hierbei sogar auf die Durchschnitts- und Komplementbildung verzichtet werden.

Definition 16. *Die Menge der* ***regulären Ausdrücke*** γ *(über einem Alphabet* Σ*) und die durch* γ *dargestellte Sprache* $L(\gamma)$ *sind induktiv wie folgt definiert. Die Symbole* $\emptyset$*,* ϵ *und* a *(*$a \in \Sigma$*) sind reguläre Ausdrücke, die*

- *die leere Sprache* $L(\emptyset) = \emptyset$*,*
- *die Sprache* $L(\epsilon) = \{\varepsilon\}$ *und*
- *für jedes Zeichen* $a \in \Sigma$ *die Sprache* $L(a) = \{a\}$

beschreiben. Sind α *und* β *reguläre Ausdrücke, die die Sprachen* $L(\alpha)$ *und* $L(\beta)$ *beschreiben, so sind auch* $\alpha\beta$*,* $(\alpha|\beta)$ *und* $(\alpha)^*$ *reguläre Ausdrücke, die die Sprachen*

- $L(\alpha\beta) = L(\alpha)L(\beta)$*,*
- $L(\alpha|\beta) = L(\alpha) \cup L(\beta)$ *und*
- $L((\alpha)^*) = L(\alpha)^*$

beschreiben.

Beispiel 17. *Die regulären Ausdrücke* ϵ^**,* $\emptyset^*$*,* $(0|1)^*00$ *und* $(\epsilon 0|\emptyset 1^*)$ *beschreiben folgende Sprachen:*

γ	ϵ^*	$\emptyset^*$	$(0\|1)^*00$	$(\epsilon 0\|\emptyset 1^*)$
$L(\gamma)$	$\{\varepsilon\}^* = \{\varepsilon\}$	$\emptyset^* = \{\varepsilon\}$	$\{x00 \mid x \in \{0,1\}^*\}$	$\{0\}$

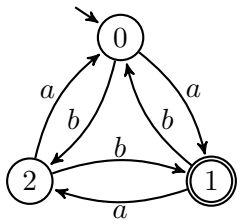
◁

Bemerkung 18.

- *Um Klammern zu sparen, definieren wir folgende* ***Präzedenzordnung****: Der Sternoperator* * *bindet stärker als der Produktoperator und dieser wiederum stärker als der Vereinigungsoperator. Für* $((a|b(c)^*)|d)$ *können wir also kurz* $a|bc^*|d$ *schreiben.*
- *Da der reguläre Ausdruck* $\gamma\gamma^*$ *die Sprache* $L(\gamma)^+$ *beschreibt, verwenden wir* γ^+ *als Abkürzung für den Ausdruck* $\gamma\gamma^*$*.*

Beispiel 19. *Betrachte nebenstehenden DFA* M*. Um für die von* M *erkannte Sprache*

$$L(M) = \{x \in \{a,b\}^* \mid \#_a(x) - \#_b(x) \equiv_3 1\}$$



einen regulären Ausdruck zu finden, betrachten wir zunächst die Sprache

$$L_0 = \{x \in \{a,b\}^* \mid \#_a(x) - \#_b(x) \equiv_3 0\}.$$

L_0 enthält also alle Wörter x, die den DFA M ausgehend vom Zustand 0 in den Zustand 0 überführen. Jedes solche x setzt sich aus beliebig vielen Teilwörtern y zusammen, die M vom Zustand 0 in den Zustand 0 überführen, ohne zwischendurch den Zustand 0 anzunehmen. Jedes solche y beginnt entweder mit einem a (Übergang von 0 nach 1) oder mit einem b (Übergang von 0 nach 2). Im ersten Fall folgt eine beliebige Anzahl von Teilwörtern ab (Wechsel zwischen 1 und 2), an die sich entweder das Suffix aa (Rückkehr von 1 nach 0 über 2) oder das Suffix b (direkte Rückkehr von 1 nach 0) anschließt.

Analog folgt im zweiten Fall eine beliebige Anzahl von Teilwörtern ba (Wechsel zwischen 2 und 1), an die sich entweder das Suffix a (direkte Rückkehr von 2 nach 0) oder das Suffix bb (Rückkehr von 2 nach 0 über 1) anschließt. Daher lässt sich L_0 durch den regulären Ausdruck

$$\gamma_0 = (a(ab)^*(aa|b) \mid b(ba)^*(a|bb))^*$$

beschreiben. Eine ähnliche Überlegung zeigt, dass sich die Wörter, die M ausgehend von 0 in den Zustand 1 überführen, ohne dass zwischendurch der Zustand 0 nochmals besucht wird, durch den regulären Ausdruck $(a|bb)(ab)^*$ beschrieben werden. Somit erhalten wir für $L(M)$ den regulären Ausdruck $\gamma = \gamma_0(a|bb)(ab)^*$. ◁

Satz 20. $\mathsf{REG} = \{L(\gamma) \mid \gamma \textit{ ist ein regulärer Ausdruck}\}$.

Beweis. Die Inklusion von rechts nach links ist klar, da die Basisausdrücke $\emptyset$, ϵ und a, $a \in \Sigma^*$, nur reguläre Sprachen beschreiben und die Sprachklasse REG unter Produkt, Vereinigung und Sternhülle abgeschlossen ist (siehe Beobachtungen 9 und 12).

Für die Gegenrichtung konstruieren wir zu einem DFA M einen regulären Ausdruck γ mit $L(\gamma) = L(M)$. Sei also $M = (Z, \Sigma, \delta, q_0, E)$ ein DFA, wobei wir annehmen können, dass $Z = \{1, \ldots, m\}$ und $q_0 = 1$ ist. Dann lässt sich $L(M)$ als Vereinigung

$$L(M) = \bigcup_{q \in E} L_{1,q}$$

von Sprachen der Form

$$L_{p,q} = \{x \in \Sigma^* \mid \hat{\delta}(p, x) = q\}$$

darstellen. Folglich reicht es zu zeigen, dass die Sprachen $L_{p,q}$ durch reguläre Ausdrücke beschreibbar sind. Hierzu betrachten wir die Sprachen

$$L^r_{p,q} = \left\{ x_1 \cdots x_n \in \Sigma^* \;\middle|\; \begin{array}{c} \hat{\delta}(p, x_1 \cdots x_n) = q \text{ und für} \\ i = 1, \ldots, n-1 \text{ gilt } \hat{\delta}(p, x_1 \cdots x_i) \leq r \end{array} \right\}.$$

Wegen $L_{p,q} = L^m_{p,q}$ reicht es, reguläre Ausdrücke $\gamma^r_{p,q}$ für die Sprachen $L^r_{p,q}$ anzugeben. Im Fall $r = 0$ enthält

$$L^0_{p,q} = \begin{cases} \{a \in \Sigma \mid \delta(p, a) = q\} \cup \{\varepsilon\}, & p = q, \\ \{a \in \Sigma \mid \delta(p, a) = q\}, & \text{sonst} \end{cases}$$

nur Buchstaben (und eventuell das leere Wort) und ist somit leicht durch einen regulären Ausdruck $\gamma^0_{p,q}$ beschreibbar. Wegen

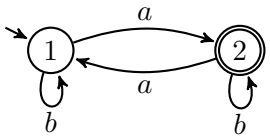
$$L^{r+1}_{p,q} \;=\; L^r_{p,q} \cup L^r_{p,r+1}(L^r_{r+1,r+1})^* L^r_{r+1,q}$$

lassen sich aus den regulären Ausdrücken $\gamma^r_{p,q}$ für die Sprachen $L^r_{p,q}$ leicht reguläre Ausdrücke für die Sprachen $L^{r+1}_{p,q}$ gewinnen:

$$\gamma^{r+1}_{p,q} = \gamma^r_{p,q} | \gamma^r_{p,r+1}(\gamma^r_{r+1,r+1})^* \gamma^r_{r+1,q}.$$

■

Beispiel 21. *Betrachte den DFA*



Da M insgesamt $m = 2$ Zustände und nur den Endzustand 2 besitzt, ist

$$L(M) = \bigcup_{q \in E} L_{1,q} = L_{1,2} = L^2_{1,2} = L(\gamma^2_{1,2}).$$

Um $\gamma^2_{1,2}$ zu berechnen, benutzen wir die Rekursionsformel

$$\gamma^{r+1}_{p,q} = \gamma^r_{p,q} | \gamma^r_{p,r+1}(\gamma^r_{r+1,r+1})^* \gamma^r_{r+1,q}$$

und erhalten

$$\begin{aligned} \gamma^2_{1,2} &= \gamma^1_{1,2} | \gamma^1_{1,2}(\gamma^1_{2,2})^* \gamma^1_{2,2}, \\ \gamma^1_{1,2} &= \gamma^0_{1,2} | \gamma^0_{1,1}(\gamma^0_{1,1})^* \gamma^0_{1,2}, \\ \gamma^1_{2,2} &= \gamma^0_{2,2} | \gamma^0_{2,1}(\gamma^0_{1,1})^* \gamma^0_{1,2}. \end{aligned}$$

Um den regulären Ausdruck $\gamma^2_{1,2}$ für $L(M)$ zu erhalten, genügt es also, die regulären Ausdrücke $\gamma^0_{1,1}$, $\gamma^0_{1,2}$, $\gamma^0_{2,1}$, $\gamma^0_{2,2}$, $\gamma^1_{1,2}$ und $\gamma^1_{2,2}$ zu berechnen:

r	p, q			
	$1,1$	$1,2$	$2,1$	$2,2$
0	$\epsilon \mid b$	a	a	$\epsilon \mid b$
1	-	$\underbrace{a \mid (\epsilon \mid b)(\epsilon \mid b)^* a}_{b^* a}$	-	$\underbrace{(\epsilon \mid b) \mid a(\epsilon \mid b)^* a}_{\epsilon \mid b \mid ab^* a}$
2	-	$\underbrace{b^* a \mid b^* a(\epsilon \mid b \mid ab^* a)^*(\epsilon \mid b \mid ab^* a)}_{b^* a(b \mid ab^* a)^*}$	-	-

◁

Korollar 22. *Sei L eine Sprache. Dann sind folgende Aussagen äquivalent:*

- *L ist regulär,*
- *es gibt einen DFA M mit $L = L(M)$,*
- *es gibt einen NFA N mit $L = L(N)$,*
- *es gibt einen regulären Ausdruck γ mit $L = L(\gamma)$,*
- *L lässt sich mit den Operationen Vereinigung, Produkt und Sternhülle aus endlichen Sprachen gewinnen,*
- *L lässt sich mit den Operationen $\cap$, $\cup$, Komplement, Produkt und Sternhülle aus endlichen Sprachen gewinnen.*

Wir werden bald noch eine weitere Charakterisierung von REG kennenlernen, nämlich durch reguläre Grammatiken. Zuvor befassen wir uns jedoch mit dem Problem, DFAs zu minimieren. Dabei spielen Relationen (insbesondere Äquivalenzrelationen) eine wichtige Rolle.

2.4 Relationalstrukturen

Sei A eine nichtleere Menge, R_i eine k_i-stellige Relation auf A, d.h. $R_i \subseteq A^{k_i}$ für $i = 1, \ldots, n$. Dann heißt $(A; R_1, \ldots, R_n)$ ***Relationalstruktur***. Die Menge A heißt ***Grundmenge***, ***Trägermenge*** oder ***Individuenbereich*** der Relationalstruktur.

Wir werden hier hauptsächlich den Fall $n = 1$, $k_1 = 2$, also (A, R) mit $R \subseteq A \times A$ betrachten. Man nennt dann R eine **(binäre) Relation** auf A. Oft wird für $(a, b) \in R$ auch die **Infix-Schreibweise** aRb benutzt.

Beispiel 23.

- (F, M) *mit* $F = \{f \mid f$ *ist Fluss in Europa*$\}$ *und*

$$M = \{(f, g) \in F \times F \mid f \text{ mündet in } g\}.$$

- (U, B) *mit* $U = \{x \mid x$ *ist Berliner*$\}$ *und*

$$B = \{(x, y) \in U \times U \mid x \text{ ist Bruder von } y\}.$$

- $(\mathsf{P}(M), \subseteq)$, *wobei* $\mathsf{P}(M)$ *die Potenzmenge einer beliebigen Menge M und $\subseteq$ die Inklusionsbeziehung auf den Teilmengen von M ist.*
- (A, Id_A), *wobei* $Id_A = \{(x, x) \mid x \in A\}$ *die* ***Identität auf*** A *ist.*
- $(\mathbb{R}, \leq)$.
- $(\mathbb{Z}, |)$, *wobei* $|$ *die "teilt"-Relation bezeichnet.*
- $(\mathcal{F}ml, \Rightarrow)$ *mit* $\mathcal{F}ml = \{F \mid F$ *ist aussagenlogische Formel*$\}$ *und*

$$\Rightarrow = \{(F, G) \in \mathcal{F}ml \times \mathcal{F}ml \mid G \text{ ist Folgerung von } F\}.$$ ◁

Da Relationen Mengen sind, sind auf ihnen die mengentheoretischen Operationen ***Durchschnitt***, ***Vereinigung***, ***Komplement***

und ***Differenz*** definiert. Seien R und S Relationen auf A, dann ist

$$\begin{aligned} R \cap S &= \{(x,y) \in A \times A \mid xRy \wedge xSy\}, \\ R \cup S &= \{(x,y) \in A \times A \mid xRy \vee xSy\}, \\ R - S &= \{(x,y) \in A \times A \mid xRy \wedge \neg xSy\}, \\ \overline{R} &= (A \times A) - R. \end{aligned}$$

Sei allgemeiner $\mathcal{M} \subseteq \mathcal{P}(A \times A)$ eine beliebige Menge von Relationen auf A. Dann sind der **Schnitt über $\mathcal{M}$** und die **Vereinigung über $\mathcal{M}$** folgende Relationen:

$$\begin{aligned} \bigcap \mathcal{M} &= \{(x,y) \mid \forall R \in \mathcal{M} : xRy\}, \\ \bigcup \mathcal{M} &= \{(x,y) \mid \exists R \in \mathcal{M} : xRy\}. \end{aligned}$$

Weiterhin ist die **Inklusionsrelation** $R \subseteq S$ auf Relationen von Bedeutung:

$$R \subseteq S \Leftrightarrow \forall x, y : xRy \rightarrow xSy.$$

Die ***transponierte (konverse) Relation*** zu R ist

$$R^T = \{(y,x) \mid xRy\}.$$

R^T wird oft auch mit R^{-1} bezeichnet. Zum Beispiel ist $(\mathbb{R}, \leq^T) = (\mathbb{R}, \geq)$.

Seien R und S Relationen auf A. Das ***Produkt*** oder die ***Komposition*** von R und S ist

$$R \circ S = \{(x,z) \in A \times A \mid \exists y \in A : xRy \wedge ySz\}.$$

Beispiel 24. *Ist B die Relation "ist Bruder von", V "ist Vater von", M "ist Mutter von" und $E = V \cup M$ "ist Elternteil von", so ist $B \circ E$ die Onkel-Relation.* ◁

Übliche Bezeichnungen für das Relationenprodukt sind auch $R;S$ und $R \cdot S$ oder einfach RS. Das n-fache Relationenprodukt $R \circ \cdots \circ R$ von R wird mit R^n bezeichnet. Dabei ist $R^0 = Id$.

Vorsicht: Das n-fache Relationenprodukt R^n von R sollte nicht mit dem n-fachen kartesischen Produkt $R \times \cdots \times R$ der *Menge* R verwechselt werden. Wir vereinbaren, dass R^n das n-fache Relationenprodukt bezeichnen soll, falls R eine Relation ist.

Eigenschaften von Relationen

Sei R eine Relation auf A. Dann heißt R

reflexiv,	falls $\forall x \in A : xRx$	(also $Id_A \subseteq R$)
irreflexiv,	falls $\forall x \in A : \neg xRx$	(also $Id_A \subseteq \overline{R}$)
symmetrisch,	falls $\forall x, y \in A : xRy \Rightarrow yRx$	(also $R \subseteq R^T$)
asymmetrisch,	falls $\forall x, y \in A : xRy \Rightarrow \neg yRx$	(also $R \subseteq \overline{R^T}$)
antisymmetrisch,	falls $\forall x, y \in A : xRy \wedge yRx \Rightarrow x = y$	(also $R \cap R^T \subseteq Id$)
konnex,	falls $\forall x, y \in A : xRy \vee yRx$	(also $A \times A \subseteq R \cup R^T$)
semikonnex,	falls $\forall x, y \in A : x \neq y \Rightarrow xRy \vee yRx$	(also $\overline{Id} \subseteq R \cup R^T$)
transitiv,	falls $\forall x, y, z \in A : xRy \wedge yRz \Rightarrow xRz$	(also $R^2 \subseteq R$)

gilt.

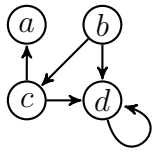
Beispiel 25.

- *Die Relation "ist Schwester von" ist zwar in einer reinen Damengesellschaft symmetrisch, i.a. jedoch weder symmetrisch noch asymmetrisch noch antisymmetrisch.*
- $(\mathbb{R}, <)$ *ist irreflexiv, asymmetrisch, transitiv und semikonnex.*
- $(\mathbb{R}, \leq)$ *und* $(\mathsf{P}(M), \subseteq)$ *sind reflexiv, antisymmetrisch und transitiv.*
- $(\mathbb{R}, \leq)$ *ist auch konnex und* $(\mathsf{P}(M), \subseteq)$ *ist im Fall* $\|M\| \leq 1$ *zwar auch konnex, aber im Fall* $\|M\| \geq 2$ *weder semikonnex noch konnex.* ◁

Graphische Darstellung von Relationen

Eine Relation R auf einer endlichen Menge A kann durch einen ***gerichteten Graphen*** (oder ***Digraphen***) $G = (V, E)$ mit ***Knotenmenge*** $V = A$ und ***Kantenmenge*** $E = R$ veranschaulicht werden. Hierzu stellen wir jedes Element $x \in A$ als einen Knoten dar und verbinden jedes Knotenpaar $(x, y) \in R$ durch eine gerichtete Kante (Pfeil). Zwei durch eine Kante verbundene Knoten heißen ***benachbart*** oder ***adjazent***.

Beispiel 26. *Für die Relation (A, R) mit $A = \{a, b, c, d\}$ und $R = \{(b, c), (b, d), (c, a), (c, d), (d, d)\}$ erhalten wir folgende graphische Darstellung.*



◁

Der ***Ausgangsgrad*** eines Knotens $x \in V$ ist $\deg^+(x) = \|R(x)\|$, wobei $R(x) = \{y \in V \mid xRy\}$ der ***Nachbereich*** von x ist. Entsprechend ist $\deg^-(x) = \|\{y \in V \mid yRx\}\|$ der ***Eingangsgrad*** von x. Falls R symmetrisch ist, werden die Pfeilspitzen meist weggelassen. In diesem Fall ist $d(x) = \deg^-(x) = \deg^+(x)$ der ***Grad*** von x. Ist R zudem irreflexiv, so ist G ***schleifenfrei*** und wir erhalten einen ***(ungerichteten) Graphen***.

Darstellung durch eine Adjazenzmatrix

Eine Relation R auf einer endlichen (geordneten) Menge $A = \{a_1, \ldots, a_n\}$ lässt sich durch eine boolesche $n \times n$-Matrix $M_R = (m_{ij})$ mit

$$m_{ij} := \begin{cases} 1, & a_i R a_j, \\ 0, & \text{sonst} \end{cases}$$

darstellen. Beispielsweise hat die Relation

$$R = \{(b, c), (b, d), (c, a), (c, d), (d, d)\}$$

auf der Menge $A = \{a, b, c, d\}$ die Matrixdarstellung

$$M_R = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Darstellung durch eine Adjazenzliste

Eine weitere Möglichkeit besteht darin, eine endliche Relation R in Form einer Tabelle darzustellen, die jedem Element $x \in A$ seinen Nachbereich $R(x)$ in Form einer Liste zuordnet:

x	$R(x)$
a	-
b	c, d
c	a, d
d	d

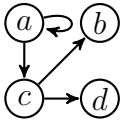
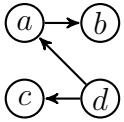
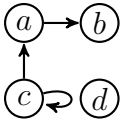
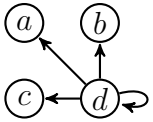
Sind $M_R = (r_{ij})$ und $M_S = (s_{ij})$ boolesche $n \times n$-Matrizen für R und S, so erhalten wir für $T = R \circ S$ die Matrix $M_T = (t_{ij})$ mit

$$t_{ij} = \bigvee_{k=1,\ldots,n} (r_{ik} \wedge s_{kj})$$

Der Nachbereich $T(x)$ von x bzgl. der Relation $T = R \circ S$ berechnet sich zu

$$T(x) = \bigcup \{S(y) \mid y \in R(x)\} = \bigcup_{y \in R(x)} S(y).$$

Beispiel 27. *Betrachte die Relationen $R = \{(a, a), (a, c), (c, b), (c, d)\}$ und $S = \{(a, b), (d, a), (d, c)\}$ auf der Menge $A = \{a, b, c, d\}$.*

Relation	R	S	$R \circ S$	$S \circ R$
Digraph				
Adjazenz-matrix	1 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0	0 1 0 0 0 0 0 0 0 0 0 0 1 0 1 0	0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
Adjazenz-liste	$a : a, c$ b : - $c : b, d$ d : -	$a : b$ b : - c : - $d : a, c$	$a : b$ b : - $c : a, c$ d : -	a : - b : - c : - $d : a, b, c, d$

◁

Beobachtung: Das Beispiel zeigt, dass das Relationenprodukt nicht kommutativ ist, d.h. i.a. gilt nicht $R \circ S = S \circ R$.

Als nächstes zeigen wir, dass die Menge $\mathcal{R} = \mathcal{P}(A \times A)$ aller binären Relationen auf A mit dem Relationenprodukt $\circ$ als binärer Operation und der Relation Id_A als neutralem Element eine Halbgruppe (oder ***Monoid***) bildet.

Satz 28. *Seien Q, R, S Relationen auf A. Dann gilt*

(i) $(Q \circ R) \circ S = Q \circ (R \circ S)$, *d.h. $\circ$ ist assoziativ,*

(ii) $Id \circ R = R \circ Id = R$, *d.h. Id ist neutrales Element.*

Beweis.

(i) Es gilt:

$$\begin{aligned} x\ (Q \circ R) \circ S\ y &\Leftrightarrow \exists u \in A : x\ (Q \circ R)\ u\ \wedge\ u\ S\ y \\ &\Leftrightarrow \exists u \in A : (\exists v \in A : x\ Q\ v\ R\ u)\ \wedge\ u\ S\ y \\ &\Leftrightarrow \exists u, v \in A : x\ Q\ v\ R\ u\ S\ y \\ &\Leftrightarrow \exists v \in A : x\ Q\ v\ \wedge\ (\exists u \in A : v\ R\ u\ \wedge\ u\ S\ y) \\ &\Leftrightarrow \exists v \in A : x\ Q\ v\ (R \circ S)\ y \\ &\Leftrightarrow x\ Q \circ (R \circ S)\ y \end{aligned}$$

(ii) Wegen $x\ Id \circ R\ y \Leftrightarrow \exists z : x = z\ \wedge\ z\ R\ y \Leftrightarrow x\ R\ y$ folgt $Id \circ R = R$. Die Gleichheit $R \circ Id = R$ folgt analog.

■

Manchmal steht man vor der Aufgabe, eine gegebene Relation R durch eine möglichst kleine Modifikation in eine Relation R' mit vorgegebenen Eigenschaften zu überführen. Will man dabei alle in R enthaltenen Paare beibehalten, dann sollte R' aus R durch Hinzufügen möglichst weniger Paare hervorgehen.

Es lässt sich leicht nachprüfen, dass der Schnitt über eine Menge reflexiver (bzw. transitiver oder symmetrischer) Relationen wieder reflexiv (bzw. transitiv oder symmetrisch) ist. Folglich existiert zu jeder Relation R auf einer Menge A eine kleinste reflexive (bzw. transitive oder symmetrische) Relation R', die R enthält.

Definition 29. *Sei R eine Relation.*

- *Die **reflexive Hülle** von R ist*

$$h_{refl}(R) = \bigcap \{S \subseteq A \times A \mid S \text{ ist reflexiv und } R \subseteq S\}.$$

- *Die **symmetrische Hülle** von R ist*

$$h_{sym}(R) = \bigcap \{S \subseteq A \times A \mid S \text{ ist symmetrisch und } R \subseteq S\}.$$

- *Die **transitive Hülle** von R ist*

$$R^+ = \bigcap \{S \subseteq A \times A \mid S \text{ ist transitiv und } R \subseteq S\}.$$

- *Die **reflexiv-transitive Hülle** von R ist*

$$R^* = \bigcap \{S \subseteq A \times A \mid S \text{ ist reflexiv, transitiv und } R \subseteq S\}.$$

Satz 30. *Sei R eine Relation auf A.*

(i) $h_{refl}(R) = R \cup Id_A$,

(ii) $h_{sym}(R) = R \cup R^T$,

(iii) $R^+ = \bigcup_{n\geq 1} R^n$,
(iv) $R^* = \bigcup_{n\geq 0} R^n$.

Beweis. Siehe Übungen. ■

Anschaulich besagt der vorhergehende Satz, dass ein Paar (a, b) genau dann in der reflexiv-transitiven Hülle R^* von R ist, wenn es ein $n \geq 0$ gibt mit aR^nb, d.h. es gibt Elemente $x_0, \ldots, x_n \in A$ mit $x_0 = a$, $x_n = b$ und

$$x_0 R x_1 R x_2 \cdots x_{n-1} R x_n.$$

In der Graphentheorie nennt man $x_0, \ldots, x_n$ einen ***Weg*** der Länge n von a nach b.

2.4.1 Äquivalenz- und Ordnungsrelationen

Die nachfolgende Tabelle gibt einen Überblick über die wichtigsten Relationalstrukturen.

	refl.	sym.	trans.	antisym.	asym.	konnex	semikon.
Äquivalenzrelation	✓	✓	✓				
(Halb-)Ordnung	✓		✓	✓			
Striktordnung			✓		✓		
lineare Ordnung			✓	✓		✓	
lin. Striktord.			✓		✓		✓
Quasiordnung	✓		✓				

In der Tabelle sind nur die definierenden Eigenschaften durch ein "✓" gekennzeichnet. Das schließt nicht aus, dass gleichzeitig auch noch weitere Eigenschaften vorliegen können.

Wir betrachten zunächst ***Äquivalenzrelationen***, die durch die drei Eigenschaften reflexiv, symmetrisch und transitiv definiert sind.

Ist E eine Äquivalenzrelation, so nennt man den zu x gehörigen Nachbereich $E(x)$ die ***von x repräsentierte Äquivalenzklasse*** und bezeichnet sie mit $[x]_E$ oder einfach mit $[x]$. Die durch E auf A induzierte Partition $\{[x] \mid x \in A\}$ wird ***Quotienten- oder Faktormenge*** genannt und mit A/E bezeichnet. Die Anzahl der Äquivalenzklassen von E wird auch als der ***Index*** von E bezeichnet. Eine Menge $S \subseteq A$ heißt ***Repräsentantensystem***, falls sie genau ein Element aus jeder Äquivalenzklasse enthält.

Beispiel 31.

- *Auf der Menge aller Geraden im $\mathbb{R}^2$ die Parallelität. Offenbar bilden alle Geraden mit derselben Richtung (oder Steigung) jeweils eine Äquivalenzklasse. Daher wird ein Repräsentantensystem beispielsweise durch die Menge aller Ursprungsgeraden gebildet.*
- *Auf der Menge aller Menschen "im gleichen Jahr geboren wie". Hier bildet jeder Jahrgang eine Äquivalenzklasse.*
- *Auf $\mathbb{Z}$ die Relation "gleicher Rest bei Division durch m". Die zugehörigen Äquivalenzklassen sind*

  $$[r] = \{a \in \mathbb{Z} \mid a \bmod m = r\}.$$

  Ein Repräsentantensystem wird also durch die Reste $\{0, 1, \ldots, m-1\}$ gebildet.
- *Auf der Menge der aussagenlogischen Formeln die semantische Äquivalenz. Hier bilden beispielsweise alle Tautologien eine Äquivalenzklasse.* ◁

Definition 32. *Eine Familie $\{M_i \mid i \in I\}$ von nichtleeren Teilmengen $M_i \subseteq A$ heißt **Partition** der Menge A, falls gilt:*

*a) die Mengen M_i **überdecken** A, d.h. $A = \bigcup_{i\in I} M_i$ und*
*b) die Mengen M_i sind **paarweise disjunkt**, d.h. für je zwei verschiedene Mengen $M_i \neq M_j$ gilt $M_i \cap M_j = \emptyset$.*

Wie der nächste Satz zeigt, beschreiben Äquivalenzrelationen auf A und Partitionen von A denselben Sachverhalt.

Satz 33. *Sei E eine Relation auf A. Dann sind folgende Aussagen äquivalent.*

(i) *E ist eine Äquivalenzrelation auf A.*

(ii) *Für alle $x, y \in A$ gilt*

$$xEy \Leftrightarrow E(x) = E(y) \qquad (*)$$

(iii) *E ist reflexiv und $\{E(x) \mid x \in A\}$ ist eine Partition von A.*

Beweis.

$(i) \Rightarrow (ii)$ Sei E eine Äquivalenzrelation auf A. Da E transitiv ist, impliziert xEy die Inklusion $E(y) \subseteq E(x)$:

$$z \in E(y) \Rightarrow yEz \Rightarrow xEz \Rightarrow z \in E(x).$$

Da E symmetrisch ist, folgt aus xEy aber auch $E(x) \subseteq E(y)$.

Umgekehrt folgt aus $E(x) = E(y)$ wegen der Reflexivität von E, dass $x \in E(x) = E(y)$ enthalten ist, und somit xEy. Dies zeigt, dass E die Äquivalenz $(*)$ erfüllt.

$(ii) \Rightarrow (iii)$ Falls E die Bedingung $(*)$ erfüllt, so folgt sofort xEx (wegen $E(x) = E(x)$) und folglich überdecken die Nachbereiche $E(x)$ (wegen $x \in E(x)$) die Menge A.

Ist $E(x) \cap E(y) \neq \emptyset$ und z ein Element in $E(x) \cap E(y)$, so gilt xEz und yEz und daher folgt $E(x) = E(z) = E(y)$. Da also je zwei Nachbereiche $E(x)$ und $E(y)$ entweder gleich oder disjunkt sind, bildet $\{E(x) \mid x \in A\}$ sogar eine Partition von A.

$(iii) \Rightarrow (i)$ Wird schließlich A von den Mengen $E(x)$ partitioniert, wobei $x \in E(x)$ für alle $x \in A$ gilt, so folgt

$$xEy \Leftrightarrow y \in E(x) \cap E(y) \Leftrightarrow E(x) = E(y).$$

Daher übertragen sich die Eigenschaften Reflexivität, Symmetrie und Transitivität unmittelbar von der Gleichheitsrelation auf E.

■

Die kleinste Äquivalenzrelation auf A ist die **Identität** Id_A, die größte die **Allrelation** $A \times A$. Die Äquivalenzklassen der Identität enthalten jeweils nur ein Element, d.h. $A/Id_A = \{\{x\} \mid x \in A\}$, und die Allrelation erzeugt nur eine Äquivalenzklasse, nämlich $A/(A \times A) = \{A\}$.

Für zwei Äquivalenzrelationen $E \subseteq E'$ sind auch die Äquivalenzklassen $[x]_E$ von E in den Klassen $[x]_{E'}$ von E' enthalten. Folglich ist jede Äquivalenzklasse von E' die Vereinigung von (evtl. mehreren) Äquivalenzklassen von E_1. Im Fall $E \subseteq E'$ sagt man auch, E_1 bewirkt eine **feinere** Partitionierung als E_2. Demnach ist die Identität die **feinste** und die Allrelation die **gröbste** Äquivalenzrelation.

Da der Schnitt über eine Menge von Äquivalenzrelationen wieder eine Äquivalenzrelation ist, können wir für eine beliebige Relation R auf einer Menge A die kleinste R umfassende Äquivalenzrelation definieren:

$$h_{\text{äq}}(R) := \bigcap \{E \mid E \text{ ist eine Äquivalenzrelation auf } A \text{ mit } R \subseteq E\}$$

In der Sprache der Graphentheorie werden die durch die Äquivalenzklassen von $h_{\text{äq}}(R)$ induzierten Teilgraphen auch die **schwachen Zusammenhangskomponenten** des Digraphen (A, R) genannt (siehe Übungen). Als nächstes betrachten wir Ordnungen.

Definition 34. *(A, R) heißt **Ordnung** (auch **Halbordnung** oder **partielle Ordnung**), wenn R eine reflexive, antisymmetrische und transitive Relation auf A ist.*

Beispiel 35.

- *$(\mathbb{Z}, \leq)$ und $(\mathbb{N}, |)$ sind Ordnungen. Erstere ist linear, letztere nicht.*

- *Für jede Menge M ist die relationale Struktur $(\mathcal{P}(M); \subseteq)$ eine Ordnung. Diese ist nur im Fall $\|M\| \leq 1$ linear.*
- *Auf der Menge $\mathcal{A}(M)$ aller Äquivalenzrelationen auf M die Relation "feiner als". Dabei ist, wie wir gesehen haben, E_1 eine Verfeinerung von E_2, falls E_1 in E_2 enthalten ist. In diesem Fall bewirkt E_1 nämlich eine feinere Klasseneinteilung auf M als E_2, da jede Äquivalenzklasse von E_1 in einer Äquivalenzklasse von E_2 enthalten ist.*
- *Ist R eine Ordnung auf A und $B \subseteq A$, so heißt die Ordnung $R_B = R \cap (B \times B)$ die **Einschränkung** (oder **Restriktion**) von R auf B. Beispielsweise ist $(\mathcal{A}(M); \subseteq)$ die Einschränkung von $(\mathcal{P}(M \times M); \subseteq)$ auf $\mathcal{A}(M)$.* ◁

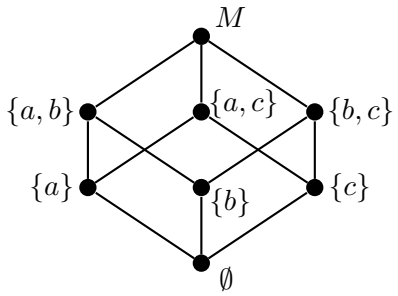
Ordnungen lassen sich sehr anschaulich durch Hasse-Diagramme darstellen. Sei $\leq$ eine Ordnung auf A und sei $<$ die Relation $\leq \cap \overline{Id}_A$. Um die Ordnung $\leq$ in einem ***Hasse-Diagramm*** darzustellen, wird nur der Graph der ***Nachbarrelation***

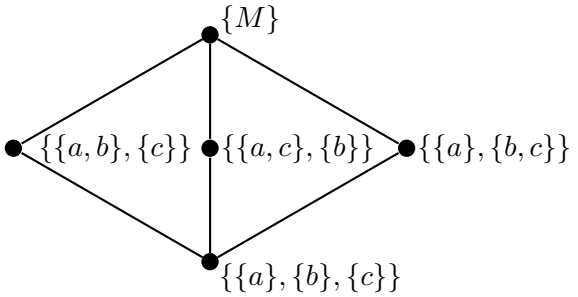
$$\lessdot \; = \; < \setminus <^2, \text{ d.h. } x \lessdot y \;\Leftrightarrow\; x < y \wedge \neg \exists z : x < z < y$$

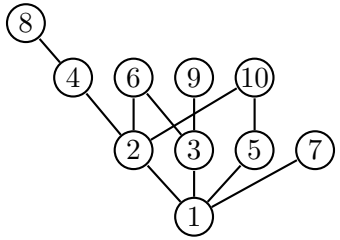
gezeichnet. Für $x \lessdot y$ sagt man auch, y ist ***oberer Nachbar*** von x. Weiterhin wird im Fall $x \lessdot y$ der Knoten y oberhalb vom Knoten x gezeichnet, so dass auf Pfeilspitzen verzichtet werden kann.

Beispiel 36.

Die Inklusionsrelation auf der Potenzmenge $\mathcal{P}(M)$ von $M = \{a, b, c\}$ lässt sich durch nebenstehendes Hasse-Diagramm darstellen.







Das linke Hasse-Diagramm stellt die "feiner als" Relation auf der Menge aller Partitionen von $M = \{a, b, c\}$ dar. Rechts ist die Einschränkung der "teilt"-Relation auf die Zahlenmenge $\{1, 2, \ldots, 10\}$ abgebildet. ◁

Definition 37. *Sei $\leq$ eine Ordnung auf A und sei b ein Element in einer Teilmenge $B \subseteq A$.*

- *b heißt **kleinstes Element** oder **Minimum** von B (kurz $b = \min B$), falls gilt:*

$$\forall b' \in B : b \leq b'.$$

- *b heißt **größtes Element** oder **Maximum** von B (kurz $b = \max B$), falls gilt:*

$$\forall b' \in B : b' \leq b.$$

- *b heißt **minimal** in B, falls es in B kein kleineres Element gibt:*

$$\forall b' \in B : b' \leq b \Rightarrow b' = b.$$

- *b heißt **maximal** in B, falls es in B kein größeres Element gibt:*

$$\forall b' \in B : b \leq b' \Rightarrow b = b'.$$

Bemerkung 38. *Da Ordnungen antisymmetrisch sind, kann es in jeder Teilmenge B höchstens ein kleinstes und höchstens ein größtes Element geben. Die Anzahl der minimalen und maximalen Elemente in B kann dagegen beliebig groß sein.*

Definition 39. *Sei $\leq$ eine Ordnung auf A und sei $B \subseteq A$.*

- *Jedes Element $u \in A$ mit $u \leq b$ für alle $b \in B$ heißt* **untere** *und jedes $o \in A$ mit $b \leq o$ für alle $b \in B$ heißt* ***obere Schranke*** *von B.*
- *B heißt* **nach oben beschränkt***, wenn B eine obere Schranke hat, und* **nach unten beschränkt***, wenn B eine untere Schranke hat.*
- *B heißt* **beschränkt***, wenn B nach oben und nach unten beschränkt ist.*
- *Besitzt B eine größte untere Schranke i, d.h. besitzt die Menge U aller unteren Schranken von B ein größtes Element i, so heißt i das* **Infimum** *von B (kurz $i = \inf B$):*

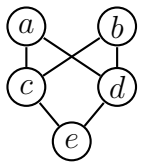
$$(\forall b \in B : b \geq i) \wedge [\forall u \in A : (\forall b \in B : b \geq u) \Rightarrow u \leq i].$$

- *Besitzt B eine kleinste obere Schranke s, d.h. besitzt die Menge O aller oberen Schranken von B ein kleinstes Element s, so heißt s das* **Supremum** *von B ($s = \sup B$):*

$$(\forall b \in B : b \leq s) \wedge [\forall o \in A : (\forall b \in B : b \leq o) \Rightarrow s \leq o]$$

Bemerkung 40. *B kann nicht mehr als ein Supremum und ein Infimum haben.*

Beispiel 41. *Betrachte nebenstehende Ordnung auf der Menge $A = \{a, b, c, d, e\}$. Die folgende Tabelle zeigt für verschiedene Teilmengen $B \subseteq A$ alle minimalen und maximalen Elemente in B Minimum und Maximum, alle unteren und oberen Schranken, sowie Infimum und Supremum von B (falls existent).*



B	*minimal*	*maximal*	*min*	*max*	*untere Schranken*	*obere Schranken*	*inf*	*sup*
$\{a, b\}$	a, b	a, b	-	-	c, d, e	-	-	-
$\{c, d\}$	c, d	c, d	-	-	e	a, b	e	-
$\{a, b, c\}$	c	a, b	c	-	c, e	-	c	-
$\{a, b, c, e\}$	e	a, b	e	-	e	-	e	-
$\{a, c, d, e\}$	e	a	e	a	e	a	e	a

◁

Bemerkung 42.

- *Auch in linearen Ordnungen muss nicht jede beschränkte Teilmenge ein Supremum oder Infimum besitzen.*
- *So hat in der linear geordneten Menge $(\mathbb{Q}, \leq)$ die Teilmenge*

$$B \;=\; \{x \in \mathbb{Q} \mid x^2 \leq 2\} \;=\; \{x \in \mathbb{Q} \mid x^2 < 2\}$$

weder ein Supremum noch ein Infimum.

- *Dagegen hat in einer linearen Ordnung jede* endliche *Teilmenge ein kleinstes und ein größtes Element und somit erst recht ein Supremum und ein Infimum.*

2.4.2 Abbildungen

Definition 43. *Sei R eine binäre Relation auf einer Menge M.*

- *R heißt* ***rechtseindeutig****, falls gilt:*

$$\forall x, y, z \in M : xRy \wedge xRz \Rightarrow y = z.$$

- *R heißt* ***linkseindeutig****, falls gilt:*

$$\forall x, y, z \in M : xRz \wedge yRz \Rightarrow x = y.$$

- *Der **Nachbereich** $N(R)$ und der **Vorbereich** $V(R)$ von R sind*
$$N(R) = \bigcup_{x \in M} R(x) \quad \text{und} \quad V(R) = \bigcup_{x \in M} R^T(x).$$
- *Eine rechtseindeutige Relation R mit $V(R) = A$ und $N(R) \subseteq B$ heißt **Abbildung** oder **Funktion von A nach B** (kurz $R : A \to B$).*

Bemerkung 44.

- *Wie üblich werden wir Abbildungen meist mit kleinen Buchstaben $f, g, h, \ldots$ bezeichnen und für $(x, y) \in f$ nicht xfy sondern $f(x) = y$ oder $f : x \mapsto y$ schreiben.*
- *Ist $f : A \to B$ eine Abbildung, so wird der Vorbereich $V(f) = A$ der **Definitionsbereich** und die Menge B der **Wertebereich** oder **Wertevorrat** von f genannt.*
- *Der Nachbereich $N(f)$ wird als **Bild** von f bezeichnet.*

Definition 45.

- *Im Fall $N(f) = B$ heißt f **surjektiv**.*
- *Ist f linkseindeutig, so heißt f **injektiv**. In diesem Fall impliziert $f(x) = f(y)$ die Gleichheit $x = y$.*
- *Eine injektive und surjektive Abbildung heißt **bijektiv**.*
- *Für eine injektive Abbildung $f : A \to B$ ist auch f^T eine Abbildung, die mit f^{-1} bezeichnet und die **inverse Abbildung** zu f genannt wird.*

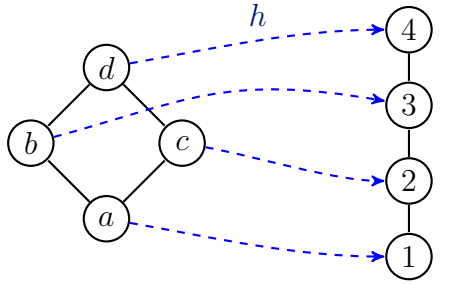
Man beachte, dass der Definitionsbereich $V(f^{-1}) = N(f)$ von f^{-1} nur dann gleich B ist, wenn f auch surjektiv, also eine Bijektion ist.

2.4.3 Homo- und Isomorphismen

Definition 46. *Seien (A_1, R_1) und (A_2, R_2) Relationalstrukturen.*

- *Eine Abbildung $h : A_1 \to A_2$ heißt **Homomorphismus**, falls für alle $a, b \in A_1$ gilt:*
$$aR_1b \Rightarrow h(a)R_2h(b).$$
- *Sind (A_1, R_1) und (A_2, R_2) Ordnungen, so spricht man von **Ordnungshomomorphismen** oder einfach von **monotonen** Abbildungen.*
- *Injektive Ordnungshomomorphismen werden auch **streng monotone** Abbildungen genannt.*

Beispiel 47. *Folgende Abbildung $h : A_1 \to A_2$ ist ein bijektiver Ordnungshomomorphismus.*



Obwohl h ein bijektiver Homomorphismus ist, ist die Umkehrung h^{-1} kein Homomorphismus, da h^{-1} nicht monoton ist. Es gilt nämlich
$$2 \sqsubseteq 3, \text{ aber } h^{-1}(2) = b \not\leq c = h^{-1}(3).$$
◁

Definition 48. *Ein bijektiver Homomorphismus $h : A_1 \to A_2$, bei dem auch h^{-1} ein Homomorphismus ist, d.h. es gilt*
$$\forall a, b \in A_1 : aR_1b \Leftrightarrow h(a)R_2h(b).$$
*heißt **Isomorphismus**. In diesem Fall heißen die Strukturen (A_1, R_1) und (A_2, R_2) **isomorph** (kurz: $(A_1, R_1) \cong (A_2, R_2)$).*

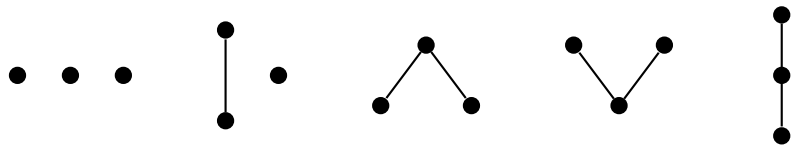
Beispiel 49.

- *Die Abbildung* $h : \mathbb{R} \to \mathbb{R}^+$ *mit*

$$h : x \mapsto e^x$$

ist ein Ordnungsisomorphismus zwischen $(\mathbb{R}, \leq)$ *und* $(\mathbb{R}^+, \leq)$.

- *Es existieren genau 5 nichtisomorphe Ordnungen mit 3 Elementen:*



Anders ausgedrückt: Die Klasse aller dreielementigen Ordnungen zerfällt unter der Äquivalenzrelation $\cong$ *in fünf Äquivalenzklassen, die durch obige fünf Hasse-Diagramme repräsentiert werden.*

- *Für* $n \in \mathbb{N}$ *sei*

$$T_n \;=\; \{k \in \mathbb{N} \mid k \text{ teilt } n\}$$

die Menge aller Teiler von n *und*

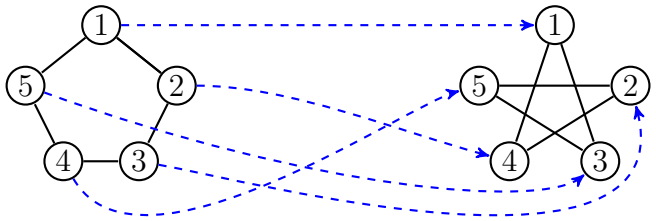
$$P_n \;=\; \{k \in \mathbb{N} \mid k \text{ ist Primteiler von } n\}$$

die Menge aller Primteiler von n. *Dann ist die Abbildung*

$$h : k \mapsto P_k$$

ein (surjektiver) Ordnungshomomorphismus von $(T_n, |)$ *auf* $(\mathcal{P}(P_n), \subseteq)$. h *ist sogar ein Isomorphismus, falls* n quadratfrei *ist (d.h. es gibt kein* $k \geq 2$, *so dass* k^2 *die Zahl* n *teilt).*

- *Die beiden folgenden Graphen* G *und* G' *sind isomorph. Zwei Isomorphismen sind beispielsweise* h_1 *und* h_2.



$G = (V, E)$ $\qquad$ $G' = (V, E')$

v	1 2 3 4 5
$h_1(v)$	1 3 5 2 4
$h_2(v)$	1 4 2 5 3

- *Während auf der Knotenmenge* $V = [3]$ *insgesamt* $2^3 = 8$ *verschiedene Graphen existieren, gibt es auf dieser Menge nur* 4 *verschiedene nichtisomorphe Graphen:*

◁

Bemerkung 50. *Auf der Knotenmenge* $V = \{1, \ldots, n\}$ *existieren genau* $2^{\binom{n}{2}}$ *verschiedene Graphen. Sei* $a(n)$ *die Anzahl aller nichtisomorphen Graphen auf* V. *Da jede Isomorphieklasse mindestens einen und höchstens* $n!$ *verschiedene Graphen enthält, ist* $2^{\binom{n}{2}}/n! \leq a(n) \leq 2^{\binom{n}{2}}$. *Tatsächlich ist* $a(n)$ ***asymptotisch gleich*** $u(n) = 2^{\binom{n}{2}}/n!$ *(in Zeichen:* $a(n) \sim u(n)$*), d.h.*

$$\lim_{n\to\infty} a(n)/u(n) = 1.$$

Also gibt es auf $V = \{1, \ldots, n\}$ *nicht wesentlich mehr als* $u(n)$ *nichtisomorphe Graphen.*

2.5 Minimierung von DFAs

Wie können wir feststellen, ob ein DFA $M = (Z, \Sigma, \delta, q_0, E)$ unnötige Zustände enthält? Zunächst einmal können alle Zustände entfernt werden, die nicht vom Startzustand aus erreichbar sind. Im folgenden gehen wir daher davon aus, dass M keine unerreichbaren Zustände enthält. Offensichtlich können zwei Zustände q und p zu einem Zustand verschmolzen werden (kurz: $q \sim p$), wenn M von q und von p ausgehend jeweils dieselben Wörter akzeptiert. Bezeichnen wir den DFA $(Z, \Sigma, \delta, q, E)$ mit M_q und $L(M_q)$ mit L_q, so sind q und p genau dann verschmelzbar, wenn $L_q = L_p$ ist.

Fassen wir alle zu einem Zustand z äquivalenten Zustände in dem neuen Zustand

$$[z]_\sim = \{z' \in Z \mid L_{z'} = L_z\}$$

zusammen (wofür wir auch kurz $[z]$ oder $\tilde{z}$ schreiben) und ersetzen wir Z und E durch $\tilde{Z} = \{\tilde{z} \mid z \in Z\}$ und $\tilde{E} = \{\tilde{z} \mid z \in E\}$, so erhalten wir den DFA $\tilde{M} = (\tilde{Z}, \Sigma, \tilde{\delta}, \tilde{q}_0, \tilde{E})$ mit

$$\tilde{\delta}(\tilde{q}, a) = \widetilde{\delta(q, a)}.$$

Hierbei bezeichnet $\tilde{Q}$ für eine Teilmenge $Q \subseteq Z$ die Menge $\{\tilde{q} \mid q \in Q\}$ aller Äquivalenzklassen $\tilde{q}$, die mindestens ein Element $q \in Q$ enthalten. Der nächste Satz zeigt, dass $\tilde{M}$ tatsächlich der gesuchte Minimalautomat ist.

Satz 51. *Sei $M = (Z, \Sigma, \delta, q_0, E)$ ein DFA, der nur Zustände enthält, die vom Startzustand q_0 aus erreichbar sind. Dann ist $\tilde{M} = (\tilde{Z}, \Sigma, \tilde{\delta}, \tilde{q}_0, \tilde{E})$ mit*

$$\tilde{\delta}(\tilde{q}, a) = \widetilde{\delta(q, a)}$$

ein DFA für $L(M)$ mit einer minimalen Anzahl von Zuständen.

Beweis. Wir zeigen zuerst, dass $\tilde{\delta}$ wohldefiniert ist, also der Wert von $\tilde{\delta}(\tilde{q}, a)$ nicht von der Wahl des Repräsentanten q abhängt. Hierzu zeigen wir, dass im Fall $p \sim q$ auch $\delta(q, a)$ und $\delta(p, a)$ äquivalent sind:

$$\begin{aligned} L_q = L_p &\Rightarrow \forall x \in \Sigma^* : x \in L_q \leftrightarrow x \in L_p \\ &\Rightarrow \forall x \in \Sigma^* : ax \in L_q \leftrightarrow ax \in L_p \\ &\Rightarrow \forall x \in \Sigma^* : x \in L_{\delta(q,a)} \leftrightarrow x \in L_{\delta(p,a)} \\ &\Rightarrow L_{\delta(q,a)} = L_{\delta(p,a)}. \end{aligned}$$

Als nächstes zeigen wir, dass $L(\tilde{M}) = L(M)$ ist. Sei $x = x_1 \cdots x_n$ eine Eingabe und seien

$$q_i = \hat{\delta}(q_0, x_1 \cdots x_i),\ i = 0, \ldots, n$$

die von M beim Abarbeiten von x durchlaufenen Zustände. Wegen

$$\tilde{\delta}(\tilde{q}_{i-1}, x_i) = \widetilde{\delta(q_{i-1}, x_i)} = \tilde{q}_i$$

durchläuft $\tilde{M}$ dann die Zustände

$$\tilde{q}_0, \tilde{q}_1, \cdots, \tilde{q}_n.$$

Da aber q_n genau dann zu E gehört, wenn $\tilde{q}_n \in \tilde{E}$ ist, folgt $L(\tilde{M}) = L(M)$ (man beachte, dass $\tilde{q}_n$ entweder nur Endzustände oder nur Nicht-Endzustände enthält, vgl. Beobachtung 52).

Es bleibt zu zeigen, dass $\tilde{M}$ eine minimale Anzahl $\|\tilde{Z}\|$ von Zuständen hat. Dies ist sicher dann der Fall, wenn bereits M minimal ist. Es reicht also zu zeigen, dass die Anzahl $k = \|\tilde{Z}\| = \|\{L_q \mid q \in Z\}\|$ der Zustände von $\tilde{M}$ nicht von M, sondern nur von $L = L(M)$ abhängt. Für $x \in \Sigma^*$ sei

$$L_x = \{y \in \Sigma^* \mid xy \in L\}.$$

Dann gilt $\{L_x \mid x \in \Sigma^*\} \subseteq \{L_q \mid q \in Z\}$, da $L_x = L_{\hat{\delta}(q_0,x)}$ ist. Die umgekehrte Inklusion gilt ebenfalls, da nach Voraussetzung jeder Zustand $q \in Z$ über ein $x \in \Sigma^*$ erreichbar ist. Also hängt $k = \|\{L_q \mid q \in Z\}\| = \|\{L_x \mid x \in \Sigma^*\}\|$ nur von L ab. ∎

Für die algorithmische Konstruktion von $\tilde{M}$ aus M ist es notwendig herauszufinden, ob zwei Zustände p und q von M äquivalent sind oder nicht.

Bezeichne $A\Delta B = (A \setminus B) \cup (B \setminus A)$ die *symmetrische Differenz* von zwei Mengen A und B. Dann ist die Inäquivalenz $p \not\sim q$ zweier Zustände p und q gleichbedeutend mit $L_p \Delta L_q \neq \emptyset$. Wir nennen ein Wort $x \in L_p \Delta L_q$ einen *Unterscheider* zwischen p und q.

Beobachtung 52.

- *Endzustände $p \in E$ sind nicht mit Zuständen $q \in Z \setminus E$ äquivalent (da sie durch ε unterschieden werden).*
- *Wenn $\delta(p,a)$ und $\delta(q,a)$ inäquivalent sind, dann auch p und q (da jeder Unterscheider x von $\delta(p,a)$ und $\delta(q,a)$ einen Unterscheider ax von p und q liefert).*

Wenn also D nur Paare von inäquivalenten Zuständen enthält, dann trifft dies auch auf die Menge

$$D' = \{\{p,q\} \mid \exists a \in \Sigma : \{\delta(p,a), \delta(q,a)\} \in D\}$$

zu. Wir können somit ausgehend von der Menge

$$D_0 = \{\{p,q\} \mid p \in E, q \notin E\}$$

eine Folge von Mengen

$$D_0 \subseteq D_1 \subseteq \cdots \subseteq \{\{z,z'\} \subseteq Z \mid z \neq z'\}$$

mittels der Vorschrift

$$D_{i+1} = D_i \cup \{\{p,q\} \mid \exists a \in \Sigma : \{\delta(p,a), \delta(q,a)\} \in D_i\}$$

berechnen, indem wir zu D_i alle Paare $\{p,q\}$ hinzufügen, für die eines der Paare $\{\delta(p,a), \delta(q,a)\}$, $a \in \Sigma$, bereits zu D_i gehört. Da Z endlich ist, muss es ein j mit $D_{j+1} = D_j$ geben. In diesem Fall gilt (siehe Übungen):

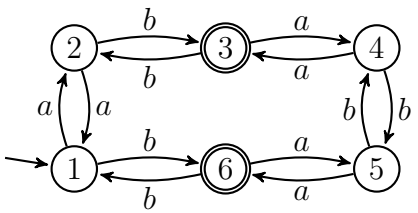
$$p \not\sim q \Leftrightarrow \{p,q\} \in D_j.$$

Folglich kann $\tilde{M}$ durch Verschmelzen aller Zustände p, q mit $\{p,q\} \notin D_j$ gebildet werden. Der folgende Algorithmus berechnet für einen beliebigen DFA M den zugehörigen Minimal-DFA $\tilde{M}$.

Algorithmus `min-DFA`(M)

```
1  Input: DFA M = (Z, Σ, δ, q_0, E)
2    entferne alle nicht erreichbaren Zustaende
3    D' := {{z, z'} | z ∈ E, z' ∉ E}
4    repeat
5      D := D'
6      D' := D ∪ {{p, q} | ∃a ∈ Σ : {δ(p, a), δ(q, a)} ∈ D}
7    until D' = D
8  Output: M̃ = (Z̃, Σ, δ̃, q̃_0, Ẽ), wobei für jeden Zustand
             z ∈ Z gilt: z̃ = {z} ∪ {z' ∈ Z | {z, z'} ∉ D}
```

Beispiel 53. *Betrachte den DFA M*



Dann enthält D_0 die Paare

$$\{1,3\}, \{1,6\}, \{2,3\}, \{2,6\}, \{3,4\}, \{3,5\}, \{4,6\}, \{5,6\}.$$

Die Paare in D_0 sind in der folgenden Matrix durch den Unterscheider

ε markiert.

2					
3	ε	ε			
4	a	a	ε		
5	a	a	ε		
6	ε	ε		ε	ε
	1	2	3	4	5

Wegen

$\{p,q\}$	$\{1,4\}$	$\{1,5\}$	$\{2,4\}$	$\{2,5\}$
$\{\delta(q,a),\delta(p,a)\}$	$\{2,3\}$	$\{2,6\}$	$\{1,3\}$	$\{1,6\}$

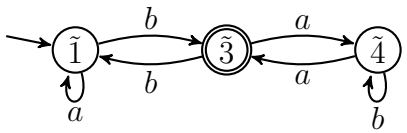
enthält D_1 zusätzlich die Paare $\{1,4\}$, $\{1,5\}$, $\{2,4\}$, $\{2,5\}$ (in obiger Matrix durch den Unterscheider a markiert). Da die verbliebenen Paare $\{1,2\}$, $\{3,6\}$, $\{4,5\}$ wegen

$\{p,q\}$	$\{1,2\}$	$\{3,6\}$	$\{4,5\}$
$\{\delta(p,a),\delta(q,a)\}$	$\{1,2\}$	$\{4,5\}$	$\{3,6\}$
$\{\delta(p,b),\delta(q,b)\}$	$\{3,6\}$	$\{1,2\}$	$\{4,5\}$

nicht zu D_1 hinzugefügt werden können, ist $D_2 = D_1$. Aus den unmarkierten Paaren $\{1,2\}$, $\{3,6\}$ und $\{4,5\}$ erhalten wir die Äquivalenzklassen

$$\tilde{1} = \{1,2\},\ \ \tilde{3} = \{3,6\}\ \ \textit{und}\ \ \tilde{4} = \{4,5\},$$

die auf folgenden Minimal-DFA $\tilde{M}$ führen:



◁

Es ist auch möglich, einen Minimalautomaten M_L direkt aus einer regulären Sprache L zu gewinnen (also ohne einen DFA M für L zu kennen). Da wegen

$$\begin{aligned}\widetilde{\hat\delta(q_0,x)} = \widetilde{\hat\delta(q_0,y)} &\Leftrightarrow \hat\delta(q_0,x) \sim \hat\delta(q_0,y)\\ &\Leftrightarrow L_{\hat\delta(q_0,x)} = L_{\hat\delta(q_0,y)} \Leftrightarrow L_x = L_y\end{aligned}$$

zwei Eingaben x und y den DFA $\tilde{M}$ genau dann in denselben Zustand $\widetilde{\hat\delta(q_0,x)} = \widetilde{\hat\delta(q_0,y)}$ überführen, wenn $L_x = L_y$ ist, können wir den von $\tilde{M}$ bei Eingabe x erreichten Zustand auch mit der Sprache L_x bezeichnen. Dies führt auf den zu $\tilde{M}$ isomorphen (also bis auf die Benennung der Zustände mit $\tilde{M}$ identischen) DFA $M_L = (Z_L, \Sigma, \delta_L, L_\varepsilon, E_L)$ mit

$$\begin{aligned} Z_L &= \{L_x \mid x \in \Sigma^*\},\\ E_L &= \{L_x \mid x \in L\} \text{ und}\\ \delta_L(L_x,a) &= L_{xa}.\end{aligned}$$

Notwendig und hinreichend für die Existenz von M_L ist, dass die Menge $\{L_x \mid x \in \Sigma^*\}$ nur endlich viele verschiedene Sprachen enthält. L ist also genau dann regulär, wenn die durch

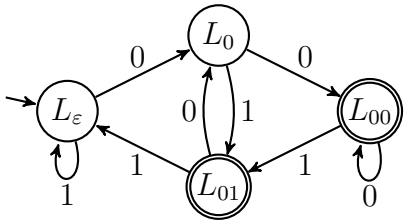
$$x\, R_L\, y \Leftrightarrow L_x = L_y$$

auf Σ^* definierte Äquivalenzrelation R_L endlichen Index hat.

Beispiel 54. *Für $L = \{x_1 \cdots x_n \in \{0,1\}^* \mid n \geq 2$ und $x_{n-1} = 0\}$ ist*

$$L_x = \begin{cases} L, & x \in \{\varepsilon, 1\} \textit{ oder } x \textit{ endet mit } 11,\\ L \cup \{0,1\}, & x = 0 \textit{ oder } x \textit{ endet mit } 10,\\ L \cup \{\varepsilon,0,1\}, & x \textit{ endet mit } 00,\\ L \cup \{\varepsilon\}, & x \textit{ endet mit } 01.\end{cases}$$

Somit erhalten wir den folgenden Minimalautomaten M_L.



◁

Im Fall, dass M bereits ein Minimalautomat ist, sind alle Zustände von $\tilde{M}$ von der Form $\tilde{q} = \{q\}$, so dass M isomorph zu $\tilde{M}$ und damit auch isomorph zu M_L ist. Dies zeigt, dass alle Minimalautomaten für eine Sprache L isomorph sind.

Satz 55 (Myhill und Nerode)**.**
Für eine Sprache L bezeichne $index(R_L) = \|\{[x]_{R_L} \mid x \in \Sigma^*\}\|$ den Index der Äquivalenzrelation R_L.

1. $\mathsf{REG} = \{L \mid index(R_L) < \infty\}$.
2. *Für jede reguläre Sprache L gibt es bis auf Isomorphie genau einen Minimal-DFA. Dieser hat $index(R_L)$ Zustände.*

Beispiel 56. *Sei $L = \{a^i b^i \mid i \geq 0\}$. Wegen $b^i \in L_{a^i} \Delta L_{a^j}$ für $i \neq j$ hat R_L unendlichen Index, d.h. L ist nicht regulär.* ◁

Die Zustände von M_L können anstelle von L_x auch mit den Äquivalenzklassen $[x]_{R_L}$ (bzw. mit geeigneten Repräsentanten) benannt werden. Der resultierende Minimal-DFA $M_{R_L} = (Z, \Sigma, \delta, [\varepsilon], E)$ mit

$$\begin{aligned} Z &= \{[x]_{R_L} \mid x \in \Sigma^*\}, \\ E &= \{[x]_{R_L} \mid x \in L\} \text{ und} \\ \delta([x]_{R_L}, a) &= [xa]_{R_L} \end{aligned}$$

wird auch als **Äquivalenzklassenautomat** bezeichnet.

Die Konstruktion von M_{R_L} ist meist einfacher als die von M_L, da die Bestimmung der Sprachen L_x entfällt. Um die Überführungsfunktion

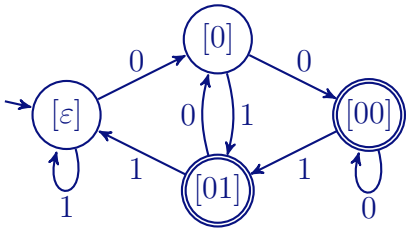
von M_{R_L} aufzustellen, reicht es, ausgehend von $r_1 = \varepsilon$ eine Folge $r_1, \ldots, r_k$ von paarweise bzgl. R_L inäquivalenten Wörtern zu bestimmen, so dass zu jedem Wort $r_i a$, $a \in \Sigma$, ein r_j mit $r_i a R_L r_j$ existiert. In diesem Fall ist $\delta([r_i], a) = [r_i a] = [r_j]$.

Beispiel 57. *Für die Sprache $L = \{x_1 \cdots x_n \in \{0,1\}^* \mid x_{n-1} = 0\}$ lässt sich M_{R_L} wie folgt konstruieren:*

1. *Wir beginnen mit $r_1 = \varepsilon$.*
2. *Da $r_1 0 = 0 \notin [\varepsilon]$ ist, wählen wir $r_2 = 0$ und setzen $\delta([\varepsilon], 0) = [0]$.*
3. *Da $r_1 1 = 1 \in [\varepsilon]$ ist, setzen wir $\delta([\varepsilon], 1) = [\varepsilon]$.*
4. *Da $r_2 0 = 00 \notin [\varepsilon] \cup [0]$ ist, ist $r_3 = 00$ und wir setzen $\delta([0], 0) = [00]$.*
5. *Da $r_2 1 = 01 \notin [\varepsilon] \cup [0] \cup [00]$ ist, wählen wir $r_4 = 01$ und setzen $\delta([0], 1) = [01]$.*
6. *Da die Wörter $r_3 0 = 000 \in [00]$, $r_3 1 = 001 \in [01]$, $r_4 0 = 010 \in [0]$ und $r_4 1 = 011 \in [\varepsilon]$ sind, setzen wir $\delta([00], 0) = [00]$, $\delta([00], 1) = [01]$, $\delta([01], 0) = [0]$ und $\delta([01], 1) = [\varepsilon]$.*

Wir erhalten also folgenden Minimal-DFA M_{R_L}:

r	ε	0	00	01
$[r0]$	$[0]$	$[00]$	$[00]$	$[0]$
$[r1]$	$[\varepsilon]$	$[01]$	$[01]$	$[\varepsilon]$



◁

Wir fassen nochmals die wichtigsten Ergebnisse zusammen.

Korollar 58. *Sei L eine Sprache. Dann sind folgende Aussagen äquivalent:*

- *L ist regulär,*

- *es gibt einen DFA M mit $L = L(M)$,*
- *es gibt einen NFA N mit $L = L(N)$,*
- *es gibt einen regulären Ausdruck γ mit $L = L(\gamma)$,*
- *die Äquivalenzrelation R_L hat endlichen Index.*

Wir werden im nächsten Abschnitt noch eine weitere Charakterisierung von REG kennenlernen, nämlich durch reguläre Grammatiken.

2.6 Grammatiken

Eine beliebte Methode, Sprachen zu beschreiben, sind Grammatiken. Implizit haben wir hiervon bei der Definition der regulären Ausdrücke bereits Gebrauch gemacht.

Beispiel 59. *Die Sprache RA aller regulären Ausdrücke über einem Alphabet $\Sigma = \{a_1, \ldots, a_k\}$ lässt sich aus dem Symbol R durch wiederholte Anwendung folgender Regeln erzeugen:*

$$\begin{array}{llll} R & \to & \emptyset, & R \to RR, \\ R & \to & \epsilon, & R \to (R|R), \\ R & \to & a_i,\ i = 1, \ldots, k, & R \to (R)^*. \end{array}$$

◁

Definition 60. *Eine **Grammatik** ist ein 4-Tupel $G = (V, \Sigma, P, S)$, wobei*

- *V eine endliche Menge von **Variablen** (auch **Nichtterminalsymbole** genannt),*
- *Σ das **Terminalalphabet**,*
- *$P \subseteq (V \cup \Sigma)^+ \times (V \cup \Sigma)^*$ eine endliche Menge von **Regeln** (oder **Produktionen**) und*
- *$S \in V$ die **Startvariable** ist.*

Für $(u, v) \in P$ schreiben wir auch kurz $u \to_G v$ bzw. $u \to v$, wenn die benutzte Grammatik aus dem Kontext ersichtlich ist.

Definition 61. *Seien $\alpha, \beta \in (V \cup \Sigma)^*$.*

*a) Wir sagen, β **ist aus** α **in einem Schritt ableitbar** (kurz: $\alpha \Rightarrow_G \beta$), falls eine Regel $u \to_G v$ und Wörter $l, r \in (V \cup \Sigma)^*$ existieren mit*

$$\alpha = lur \text{ und } \beta = lvr.$$

Hierfür schreiben wir auch $l\underline{u}r \Rightarrow_G lvr$. (Man beachte, dass durch Unterstreichen von u in α sowohl die benutzte Regel als auch die Stelle in α, an der u durch v ersetzt wird, eindeutig erkennbar sind.)

*b) Eine Folge $\sigma = (l_0, u_0, r_0), \ldots, (l_m, u_m, r_m)$ von Tripeln (l_i, u_i, r_i) heißt **Ableitung von** β **aus** α, falls gilt:*

- *$l_0u_0r_0 = \alpha$, $l_mu_mr_m = \beta$ und*
- *$l_i\underline{u_i}r_i \Rightarrow l_{i+1}u_{i+1}r_{i+1}$ für $i = 0, \ldots, m-1$.*

*Die **Länge** von σ ist m und wir notieren σ auch in der Form*

$$l_0\underline{u_0}r_0 \Rightarrow l_1\underline{u_1}r_1 \Rightarrow \cdots \Rightarrow l_{m-1}\underline{u_{m-1}}r_{m-1} \Rightarrow l_mu_mr_m.$$

*c) Die durch G **erzeugte Sprache** ist*

$$L(G) = \{x \in \Sigma^* \mid S \Rightarrow_G^* x\}.$$

*d) Ein Wort $\alpha \in (V \cup \Sigma)^*$ mit $S \Rightarrow_G^* \alpha$ heißt **Satzform** von G.*

Zur Erinnerung: Die Relation $\Rightarrow^*$ bezeichnet die reflexive, transitive Hülle der Relation $\Rightarrow$, d.h. $\alpha \Rightarrow^* \beta$ bedeutet, dass es ein $n \geq 0$ gibt mit $\alpha \Rightarrow^n \beta$. Hierzu sagen wir auch, β **ist aus** α **(in** n **Schritten) ableitbar**. Die Relation $\Rightarrow^n$ bezeichnet das n-fache Produkt der Relation $\Rightarrow$, d.h. es gilt $\alpha \Rightarrow^n \beta$, falls Wörter $\alpha_0, \ldots, \alpha_n$ existieren mit

- $\alpha_0 = \alpha$, $\alpha_n = \beta$ und
- $\alpha_i \Rightarrow \alpha_{i+1}$ für $i = 0, \ldots, n-1$.

Beispiel 62. *Wir betrachten nochmals die Grammatik* $G = (\{R\}, \Sigma \cup \{\emptyset, \epsilon, (,), ^*, |\}, P, R)$*, die die Menge der regulären Ausdrücke über dem Alphabet* Σ *erzeugt, wobei* P *die oben angegebenen Regeln enthält. Ist* $\Sigma = \{0, 1\}$*, so lässt sich der reguläre Ausdruck* $(01)^*(\epsilon|\emptyset)$ *beispielsweise wie folgt ableiten:*

$$\begin{aligned}\underline{R} &\Rightarrow \underline{R}R \Rightarrow (\underline{R})^*R \Rightarrow (RR)^*R \Rightarrow (\underline{R}R)^*(R|R)\\ &\Rightarrow (0\underline{R})^*(R|R) \Rightarrow (01)^*(\underline{R}|R) \Rightarrow (01)^*(\epsilon|\underline{R}) \Rightarrow (01)^*(\epsilon|\emptyset)\end{aligned}$$

◁

Man unterscheidet vier verschiedene Typen von Grammatiken.

Definition 63. *Sei* $G = (V, \Sigma, P, S)$ *eine Grammatik.*

1. *G heißt* ***vom Typ 3*** *oder* ***regulär****, falls für alle Regeln* $u \to v$ *gilt:* $u \in V$ *und* $v \in \Sigma V \cup \Sigma \cup \{\epsilon\}$*.*
2. *G heißt* ***vom Typ 2*** *oder* ***kontextfrei****, falls für alle Regeln* $u \to v$ *gilt:* $u \in V$*.*
3. *G heißt* ***vom Typ 1*** *oder* ***kontextsensitiv****, falls für alle Regeln* $u \to v$ *gilt:* $|v| \geq |u|$ *(mit Ausnahme der* ε*-Sonderregel, siehe unten).*
4. *Jede Grammatik ist automatisch* ***vom Typ 0****.*

ε-Sonderregel: In einer kontextsensitiven Grammatik $G = (V, \Sigma, P, S)$ kann auch die Regel $S \to \varepsilon$ benutzt werden. Aber nur, wenn das Startsymbol S nicht auf der rechten Seite einer Regel in P vorkommt.

Die Sprechweisen „vom Typ i“ bzw. „regulär“, „kontextfrei“ und „kontextsensitiv“ werden auch auf die durch solche Grammatiken erzeugte Sprachen angewandt. (Der folgende Satz rechtfertigt dies für die regulären Sprachen, die wir bereits mit Hilfe von DFAs definiert haben.) Die zugehörigen neuen Sprachklassen sind

$$\mathsf{CFL} = \{L(G) \mid G \text{ ist eine kontextfreie Grammatik}\},$$

(*context free languages*) und

$$\mathsf{CSL} = \{L(G) \mid G \text{ ist eine kontextsensitive Grammatik}\}$$

(*context sensitive languages*). Da die Klasse der Typ 0 Sprachen mit der Klasse der rekursiv aufzählbaren (*recursively enumerable*) Sprachen übereinstimmt, bezeichnen wir diese Sprachklasse mit

$$\mathsf{RE} = \{L(G) \mid G \text{ ist eine Grammatik}\}.$$

Die Sprachklassen

$$\mathsf{REG} \subset \mathsf{CFL} \subset \mathsf{CSL} \subset \mathsf{RE}$$

bilden eine Hierarchie (d.h. alle Inklusionen sind echt), die so genannte **Chomsky-Hierarchie**.

Als nächstes zeigen wir, dass sich mit regulären Grammatiken gerade die regulären Sprachen erzeugen lassen. Hierbei erweist sich folgende Beobachtung als nützlich.

Lemma 64. *Zu jeder regulären Grammatik* $G = (V, \Sigma, P, S)$ *gibt es eine äquivalente reguläre Grammatik* G'*, die keine Produktionen der Form* $A \to a$ *hat.*

Beweis. Betrachte die Grammatik $G' = (V', \Sigma, P', S)$ mit

$$\begin{aligned}V' &= V \cup \{X_{neu}\},\\ P' &= \{A \to aX_{neu} \mid A \to_G a\} \cup \{X_{neu} \to \varepsilon\} \cup P \setminus (V \times \Sigma).\end{aligned}$$

Es ist leicht zu sehen, dass G' die gleiche Sprache wie G erzeugt. ■

Satz 65. $\mathsf{REG} = \{L(G) \mid G \text{ ist eine reguläre Grammatik}\}$.

Beweis. Sei $L \in \mathsf{REG}$ und sei $M = (Z, \Sigma, \delta, q_0, E)$ ein DFA mit $L(M) = L$. Wir konstruieren eine reguläre Grammatik $G = (V, \Sigma, P, S)$ mit $L(G) = L$. Setzen wir

$$\begin{aligned}V &= Z,\\ S &= q_0 \text{ und}\\ P &= \{q \to ap \mid \delta(q, a) = p\} \cup \{q \to \varepsilon \mid q \in E\},\end{aligned}$$

so gilt für alle Wörter $x = x_1 \cdots x_n \in \Sigma^*$:

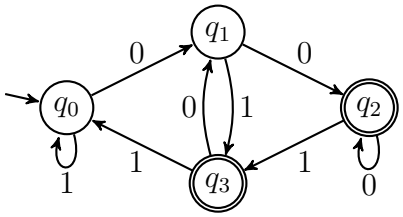
$$\begin{aligned} x \in L(M) &\Leftrightarrow \exists\, q_1, \ldots, q_{n-1} \in Z\, \exists\, q_n \in E : \\ &\quad \delta(q_{i-1}, x_i) = q_i \text{ für } i = 1, \ldots, n \\ &\Leftrightarrow \exists\, q_1, \ldots, q_n \in V : \\ &\quad q_{i-1} \to_G x_i q_i \text{ für } i = 1, \ldots, n \text{ und } q_n \to_G \varepsilon \\ &\Leftrightarrow \exists\, q_1, \ldots, q_n \in V : \\ &\quad q_0 \Rightarrow_G^i x_1 \cdots x_i q_i \text{ für } i = 1, \ldots, n \text{ und } q_n \to_G \varepsilon \\ &\Leftrightarrow x \in L(G) \end{aligned}$$

Für die entgegengesetzte Inklusion sei nun $G = (V, \Sigma, P, S)$ eine reguläre Grammatik, die keine Produktionen der Form $A \to a$ enthält. Dann können wir die gerade beschriebene Konstruktion einer Grammatik aus einem DFA „umdrehen“, um ausgehend von G einen NFA $M = (Z, \Sigma, \delta, \{S\}, E)$ mit

$$\begin{aligned} Z &= V, \\ E &= \{A \mid A \to_G \varepsilon\} \text{ und} \\ \delta(A, a) &= \{B \mid A \to_G aB\} \end{aligned}$$

zu erhalten. Genau wie oben folgt nun $L(M) = L(G)$. ■

Beispiel 66. *Der DFA*



führt auf die Grammatik $(\{q_0, q_1, q_2, q_3\}, \{0, 1\}, P, q_0)$ *mit*

$$\begin{aligned} P: \quad & q_0 \to 1q_0, 0q_1, \\ & q_1 \to 0q_2, 1q_3, \\ & q_2 \to 0q_2, 1q_3, \varepsilon, \\ & q_3 \to 0q_1, 1q_0, \varepsilon. \end{aligned}$$

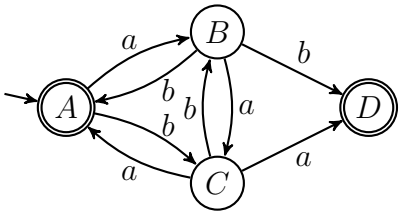
Umgekehrt führt die Grammatik $G = (\{A, B, C\}, \{a, b\}, P, A)$ *mit*

$$\begin{aligned} P: \quad & A \to aB, bC, \varepsilon, \\ & B \to aC, bA, b, \\ & C \to aA, bB, a \end{aligned}$$

über die Grammatik $G' = (\{A, B, C, D\}, \{a, b\}, P', A)$ *mit*

$$\begin{aligned} P': \quad & A \to aB, bC, \varepsilon, \\ & B \to aC, bA, bD, \\ & C \to aA, bB, aD, \\ & D \to \varepsilon \end{aligned}$$

auf den NFA



◁

2.7 Das Pumping-Lemma

Wie kann man von einer Sprache nachweisen, dass sie nicht regulär ist? Eine Möglichkeit besteht darin, die Kontraposition folgender Aussage anzuwenden.

Satz 67 (Pumping-Lemma für reguläre Sprachen)**.**
Zu jeder regulären Sprache L gibt es eine Zahl l, so dass sich alle Wörter $x \in L$ mit $|x| \geq l$ in $x = uvw$ zerlegen lassen mit

1. *$v \neq \varepsilon$,*
2. *$|uv| \leq l$ und*
3. *$uv^i w \in L$ für alle $i \geq 0$.*

*Falls eine Zahl l mit diesen Eigenschaften existiert, wird das kleinste solche l die **Pumping-Zahl** von L genannt.*

Beweis. Sei $G = (V, \Sigma, P, S)$ eine reguläre Grammatik für L, die keine Regeln der Form $A \to a$ enthält, und sei

$$\underline{A_0} \Rightarrow x_1\underline{A_1} \Rightarrow x_1x_2\underline{A_2} \Rightarrow \cdots \Rightarrow x_1x_2\cdots x_n\underline{A_n} \Rightarrow x_1x_2\cdots x_n$$

eine beliebige Ableitung von $x = x_1 \cdots x_n \in L$ aus $A_0 = S$. Setzen wir $l = \|V\|$, so muss im Fall $|x| = n \geq l$ unter $A_0, \ldots, A_l$ eine Variable A mehrfach vorkommen, d.h. es ex. $0 \leq j < k \leq l$ mit $A_j = A_k = A$. Somit können wir die Ableitung von x wie folgt zerlegen:

$$A_0 \Rightarrow^j x_1 \cdots x_j A_j = uA \Rightarrow^{k-j} ux_{j+1} \cdots x_k A_k = uvA \Rightarrow^{n+1-k} uvw,$$

wobei $u = x_1 \cdots x_j$, $v = x_{j+1} \cdots x_k$ und $w = x_{k+1} \cdots x_n$ ist. Dann gilt $|v| = k - j \geq 1$ (d.h. $v \neq \varepsilon$), $k = |uv| \leq l$ und für $i \geq 0$ zeigt die Ableitung

$$A_0 \Rightarrow^j uA \Rightarrow^{(k-j)i} uv^iA \Rightarrow^{n+1-k} uv^iw,$$

dass $uv^iw \in L$ ist.

Das Pumping-Lemma lässt sich alternativ unter Benutzung eines DFA $M = (Z, \Sigma, \delta, q_0, E)$ für L beweisen. Ist l die Anzahl der Zustände von M und setzen wir M auf eine Eingabe $x = x_1 \cdots x_n \in L$ der Länge $n \geq l$ an, so muss M nach spätestens l Schritten einen Zustand $q \in Z$ zum zweiten Mal annehmen:

$$\exists j, k : 0 \leq j < k \leq l \wedge \hat{\delta}(q_0, x_1 \cdots x_j) = \hat{\delta}(q_0, x_1 \cdots x_k) = q.$$

Wählen wir nun $u = x_1 \cdots x_j$, $v = x_{j+1} \cdots x_k$ und $w = x_{k+1} \cdots x_n$, so ist $|v| = k - j \geq 1$ und $|uv| = k \leq l$. Ausserdem gilt $uv^iw \in L$ für $i \geq 0$, da wegen $\hat{\delta}(q, v) = q$

$$\hat{\delta}(q_0, uv^iw) = \hat{\delta}(\hat{\delta}(\underbrace{\hat{\delta}(q_0, u)}_{q}, v^i), w) = \hat{\delta}(\underbrace{\hat{\delta}(q, v^i)}_{q}, w) = \hat{\delta}(q_0, x) \in E$$

ist. ■

Beispiel 68. *Die Sprache*

$$L = \{x \in \{a, b\}^* \mid \#_a(x) - \#_b(x) \equiv_3 1\}$$

hat die Pumping-Zahl $l = 3$. Sei nämlich $x \in L$ beliebig mit $|x| \geq 3$. Dann lässt sich innerhalb des Präfixes von x der Länge drei ein nichtleeres Teilwort v finden, das gepumpt werden kann:

1. Fall: *x hat das Präfix ab (oder ba).*
Zerlege $x = uvw$ mit $u = \varepsilon$ und $v = ab$ (bzw. $v = ba$).

2. Fall: *x hat das Präfix aab (oder bba).*
Zerlege $x = uvw$ mit $u = a$ (bzw. $u = b$) und $v = ab$ (bzw. $v = ba$).

3. Fall: *x hat das Präfix aaa (oder bbb).*
Zerlege $x = uvw$ mit $u = \varepsilon$ und $v = aaa$ (bzw. $v = bbb$). ◁

Beispiel 69. *Eine endliche Sprache L hat die Pumping-Zahl*

$$l = \begin{cases} 0, & L = \emptyset, \\ \max\{|x| + 1 \mid x \in L\}, & \text{sonst.} \end{cases}$$

Tatsächlich lässt sich jedes Wort $x \in L$ der Länge $|x| \geq l$ „pumpen“ (da solche Wörter gar nicht existieren), weshalb die Pumping-Zahl höchstens l ist. Zudem gibt es im Fall $l > 0$ ein Wort $x \in L$ der Länge $|x| = l - 1$, das sich nicht „pumpen“ lässt, weshalb die Pumping-Zahl nicht kleiner als l sein kann. ◁

Wollen wir mit Hilfe des Pumping-Lemmas von einer Sprache L zeigen, dass sie nicht regulär ist, so genügt es, für jede Zahl l ein Wort $x \in L$ der Länge $|x| \geq l$ anzugeben, so dass für jede Zerlegung von x in drei Teilwörter u, v, w mindestens eine der drei in Satz 67 aufgeführten Eigenschaften verletzt ist.

Beispiel 70. *Die Sprache*

$$L = \{a^j b^j \mid j \geq 0\}$$

ist nicht regulär, da sich für jede Zahl $l \geq 0$ das Wort $x = a^l b^l$ der Länge $|x| = 2l \geq l$ in der Sprache L befindet, welches offensichtlich nicht in Teilwörter u, v, w mit $v \neq \varepsilon$ und $uv^2w \in L$ zerlegbar ist. ◁

Beispiel 71. *Die Sprache*

$$L = \{a^{n^2} \mid n \geq 0\}$$

ist ebenfalls nicht regulär. Andernfalls müsste es nämlich eine Zahl l geben, so dass jede Quadratzahl $n^2 \geq l$ als Summe von natürlichen Zahlen $u + v + w$ darstellbar ist mit der Eigenschaft, dass $v \geq 1$ und $u + v \leq l$ ist, und für jedes $i \geq 0$ auch $u + iv + w$ eine Quadratzahl ist. Insbesondere müsste also $u + 2v + w = n^2 + v$ eine Quadratzahl sein, was wegen

$$n^2 < n^2 + v \leq n^2 + l < n^2 + 2l + 1 = (n + 1)^2$$

ausgeschlossen ist. ◁

Beispiel 72. *Auch die Sprache*

$$L = \{a^p \mid p \text{ prim } \}$$

ist nicht regulär, da sich sonst jede Primzahl p einer bestimmten Mindestgröße l als Summe von natürlichen Zahlen $u + v + w$ darstellen ließe, so dass $v \geq 1$ und für alle $i \geq 0$ auch $u + iv + w = p + (i - 1)v$ prim ist. Dies ist jedoch für $i = p + 1$ wegen

$$p + (p + 1 - 1)v = p(1 + v)$$

nicht der Fall. ◁

Bemerkung 73. *Mit Hilfe des Pumping-Lemmas kann nicht für jede Sprache $L \notin \mathsf{REG}$ gezeigt werden, dass L nicht regulär ist, da seine Umkehrung falsch ist. So hat beispielsweise die Sprache*

$$L = \{a^i b^j c^k \mid i = 0 \text{ oder } j = k\}$$

die Pumping-Zahl 1 (d.h. jedes Wort $x \in L$ mit Ausnahme von ε kann „gepumpt“ werden). Dennoch ist L nicht regulär (siehe Übungen).

3 Kontextfreie Sprachen

Wie wir gesehen haben, ist die Sprache $L = \{a^n b^n \mid n \geq 0\}$ nicht regulär. Es ist aber leicht, eine kontextfreie Grammatik für L zu finden:

$$G = (\{S\}, \{a, b\}, \{S \to aSb, S \to \varepsilon\}, S).$$

Damit ist klar, dass die Klasse der regulären Sprachen echt in der Klasse der kontextfreien Sprachen enthalten ist. Als nächstes wollen wir zeigen, dass die Klasse der kontextfreien Sprachen wiederum echt in der Klasse der kontextsensitiven Sprachen enthalten ist:

$$\mathsf{REG} \subsetneq \mathsf{CFL} \subsetneq \mathsf{CSL}.$$

Kontextfreie Grammatiken sind dadurch charakterisiert, dass sie nur Regeln der Form $A \to \alpha$ haben. Dies lässt die Verwendung von beliebigen ε-Regeln der Form $A \to \varepsilon$ zu. Eine kontextsensitive Grammatik darf dagegen höchstens die ε-Regel $S \to \varepsilon$ haben. Voraussetzung hierfür ist, dass S das Startsymbol ist und dieses nicht auf der rechten Seite einer Regel vorkommt. Daher sind nicht alle kontextfreien Grammatiken kontextsensitiv. Es lässt sich jedoch zu jeder kontextfreien Grammatik eine äquivalente kontextfreie Grammatik G' konstruieren, die auch kontextsensitiv ist. Hierzu zeigen wir zuerst, dass sich zu jeder kontextfreien Grammatik G, in der nicht das leere Wort ableitbar ist, eine äquivalente kontextfreie Grammatik G' ohne ε-Regeln konstruieren lässt.

Satz 74. *Zu jeder kontextfreien Grammatik G gibt es eine kontextfreie Grammatik G' ohne ε-Produktionen mit $L(G') = L(G) \setminus \{\varepsilon\}$.*

Beweis. Zuerst sammeln wir mit folgendem Algorithmus alle Variablen A, aus denen das leere Wort ableitbar ist. Diese werden auch als *ε-ableitbar* bezeichnet.

```
1  E' := {A ∈ V | A → ε}
2  repeat
3     E := E'
4     E' := E ∪ {A ∈ V | ∃ B_1, ..., B_k ∈ E : A → B_1 ··· B_k}
5  until E = E'
```

Nun konstruieren wir $G' = (V, \Sigma, P', S)$ wie folgt:

> Nehme zu P' alle Regeln $A \to \alpha'$ mit $\alpha' \neq \varepsilon$ hinzu, für die P eine Regel $A \to \alpha$ enthält, so dass α' aus α durch Entfernen von beliebig vielen Variablen $A \in E$ hervorgeht.

■

Beispiel 75. *Betrachte die Grammatik $G = (V, \Sigma, P, S)$ mit $V = \{S, T, U, X, Y, Z\}$, $\Sigma = \{a, b, c\}$ und den Regeln*

$$\begin{aligned} P: \quad & S \to aY, bX, Z; \quad Y \to bS, aYY; \quad T \to U; \\ & X \to aS, bXX; \quad Z \to \varepsilon, S, T, cZ; \quad U \to abc. \end{aligned}$$

Bei der Berechnung von $E = \{A \in V \mid A \Rightarrow^* \varepsilon\}$ ergeben sich der Reihe nach folgende Belegungen für die Mengenvariablen E und E':

E'	$\{Z\}$	$\{Z, S\}$
E	$\{Z, S\}$	$\{Z, S\}$

Um nun die Regelmenge P' zu bilden, entfernen wir aus P die einzige ε-Regel $Z \to \varepsilon$ und fügen die Regeln $X \to a$ (wegen $X \to aS$), $Y \to b$ (wegen $Y \to bS$) und $Z \to c$ (wegen $Z \to cZ$) hinzu:

$$\begin{aligned} P': \quad & S \to aY, bX, Z; \quad Y \to b, bS, aYY; \quad T \to U; \\ & X \to a, aS, bXX; \quad Z \to c, S, T, cZ; \quad U \to abc. \end{aligned}$$ ◁

Als direkte Anwendung des obigen Satzes können wir die Inklusion der Klasse der Typ 2 Sprachen in der Klasse der Typ 1 Sprachen zeigen.

Korollar 76. $\mathsf{REG} \subsetneq \mathsf{CFL} \subseteq \mathsf{CSL} \subseteq \mathsf{RE}$.

Beweis. Die Inklusionen $\mathsf{REG} \subseteq \mathsf{CFL}$ und $\mathsf{CSL} \subseteq \mathsf{RE}$ sind klar. Wegen $\{a^n b^n | n \geq 0\} \in \mathsf{CFL} - \mathsf{REG}$ ist die Inklusion $\mathsf{REG} \subseteq \mathsf{CFL}$ auch echt. Also ist nur noch die Inklusion $\mathsf{CFL} \subseteq \mathsf{CSL}$ zu zeigen. Nach obigem Satz ex. zu $L \in \mathsf{CFL}$ eine kontextfreie Grammatik $G = (V, \Sigma, P, S)$ ohne ε-Produktionen mit $L(G) = L \setminus \{\varepsilon\}$. Da G dann auch kontextsensitiv ist, folgt hieraus im Fall $\varepsilon \notin L$ unmittelbar $L(G) = L \in \mathsf{CSL}$. Im Fall $\varepsilon \in L$ erzeugt die kontextsensitive Grammatik

$$G' = (V \cup \{S'\}, \Sigma, P \cup \{S' \to S, \varepsilon\}, S')$$

die Sprache $L(G') = L$, d.h. $L \in \mathsf{CSL}$. ■

Als nächstes zeigen wir folgende Abschlusseigenschaften der kontextfreien Sprachen.

Satz 77. *Die Klasse* CFL *ist abgeschlossen unter Vereinigung, Produkt und Sternhülle.*

Beweis. Seien $G_i = (V_i, \Sigma, P_i, S_i)$, $i = 1, 2$, kontextfreie Grammatiken für die Sprachen $L(G_i) = L_i$ mit $V_1 \cap V_2 = \emptyset$ und sei S eine neue Variable. Dann erzeugt die kontextfreie Grammatik

$$G_3 = (V_1 \cup V_2 \cup \{S\}, \Sigma, P_1 \cup P_2 \cup \{S \to S_1, S_2\}, S)$$

die Vereinigung $L(G_3) = L_1 \cup L_2$. Die Grammatik

$$G_4 = (V_1 \cup V_2 \cup \{S\}, \Sigma, P_1 \cup P_2 \cup \{S \to S_1 S_2\}, S)$$

erzeugt das Produkt $L(G_4) = L_1 L_2$ und die Sternhülle $(L_1)^*$ wird von der Grammatik

$$G_5 = (V_1 \cup \{S\}, \Sigma, P_1 \cup \{S \to S_1 S, \varepsilon\}, S)$$

erzeugt. ■

Offen bleibt zunächst, ob die kontextfreien Sprachen auch unter Durchschnitt und Komplement abgeschlossen sind. Hierzu müssen wir für bestimmte Sprachen nachweisen, dass sie nicht kontextfrei sind. Dies gelingt mit einem Pumping-Lemma für kontextfreie Sprachen, für dessen Beweis wir Grammatiken in Chomsky-Normalform benötigen.

3.1 Chomsky-Normalform

Definition 78. *Eine Grammatik* (V, Σ, P, S) *ist in* ***Chomsky-Normalform*** *(**CNF**), falls* $P \subseteq V \times (V^2 \cup \Sigma)$ *ist, also alle Regeln die Form* $A \to BC$ *oder* $A \to a$ *haben.*

Um eine kontextfreie Grammatik in Chomsky-Normalform zu bringen, müssen wir neben den ε-Regeln $A \to \varepsilon$ auch sämtliche Variablenumbenennungen $A \to B$ loswerden.

Definition 79. *Regeln der Form* $A \to B$ *heißen* ***Variablenumbenennungen****.*

Satz 80. *Zu jeder kontextfreien Grammatik* G *ex. eine kontextfreie Grammatik* G' *ohne Variablenumbenennungen mit* $L(G') = L(G)$*.*

Beweis. Zuerst entfernen wir sukzessive alle Zyklen

$$A_1 \to A_2 \to \cdots \to A_k \to A_1,$$

indem wir diese Regeln aus P entfernen und alle übrigen Vorkommen der Variablen $A_2, \ldots, A_k$ durch A_1 ersetzen. Falls sich unter den entfernten Variablen $A_2, \ldots, A_k$ die Startvariable S befindet, sei A_1 die neue Startvariable.

Nun entfernen wir sukzessive die restlichen Variablenumbenennungen, indem wir

- eine Regel $A \to B$ wählen, so dass in P keine Variablenumbenennung $B \to C$ mit B auf der rechten Seite existiert,
- diese Regel $A \to B$ aus P entfernen und

- für jede Regel $B \to \alpha$ in P die Regel $A \to \alpha$ zu P hinzunehmen.

■

Beispiel 81. *Ausgehend von den Produktionen*

$$P\colon S \to aY, bX, Z; \quad Y \to b, bS, aYY; \quad T \to U;$$
$$X \to a, aS, bXX; \quad Z \to c, S, T, cZ; \quad U \to abc$$

entfernen wir den Zyklus $S \to Z \to S$, indem wir die Regeln $S \to Z$ und $Z \to S$ entfernen und dafür die Produktionen $S \to c, T, cS$ (wegen $Z \to c, T, cZ$) hinzunehmen:

$$S \to aY, bX, c, T, cS; \quad Y \to b, bS, aYY; \quad T \to U;$$
$$X \to a, aS, bXX; \quad U \to abc.$$

Nun entfernen wir die Regel $T \to U$ und fügen die Regel $T \to abc$ (wegen $U \to abc$) hinzu:

$$S \to aY, bX, c, T, cS; \quad Y \to b, bS, aYY; \quad T \to abc;$$
$$X \to a, aS, bXX; \quad U \to abc.$$

Als nächstes entfernen wir dann auch die Regel $S \to T$ und fügen die Regel $S \to abc$ (wegen $T \to abc$) hinzu:

$$S \to abc, aY, bX, c, cS; \quad Y \to b, bS, aYY; \quad T \to abc;$$
$$X \to a, aS, bXX; \quad U \to abc.$$

Da T und U nun nirgends mehr auf der rechten Seite vorkommen, können wir die Regeln $T \to abc$ und $U \to abc$ weglassen:

$$S \to abc, aY, bX, c, cS;\ Y \to b, bS, aYY;\ X \to a, aS, bXX.$$

◁

Nach diesen Vorarbeiten ist es nun leicht, eine gegebene kontextfreie Grammatik in Chomsky-Normalform umzuwandeln.

Satz 82. *Zu jeder kontextfreien Sprache $L \in \mathsf{CFL}$ gibt es eine CNF-Grammatik G' mit $L(G') = L \setminus \{\varepsilon\}$.*

Beweis. Aufgrund der beiden vorigen Sätze hat $L \setminus \{\varepsilon\}$ eine kontextfreie Grammatik $G = (V, \Sigma, P, S)$ ohne ε-Produktionen und ohne Variablenumbenennungen. Wir transformieren G wie folgt in eine CNF-Grammatik.

- Füge für jedes Terminalsymbol $a \in \Sigma$ eine neue Variable X_a zu V und eine neue Regel $X_a \to a$ zu P hinzu.
- Ersetze alle Vorkommen von a durch X_a, außer wenn a alleine auf der rechten Seite einer Regel steht.
- Ersetze jede Regel $A \to B_1 \cdots B_k$, $k \geq 3$, durch die $k-1$ Regeln

  $$A \to B_1A_1,\ A_1 \to B_2A_2,\ \ldots,\ A_{k-3} \to B_{k-2}A_{k-2},\ A_{k-2} \to B_{k-1}B_k,$$

  wobei $A_1, \ldots, A_{k-2}$ neue Variablen sind. ■

Beispiel 83. *In der Produktionenmenge*

$$P\colon\ S \to abc, aY, bX, c, cS;\ \ X \to a, aS, bXX;\ \ Y \to b, bS, aYY$$

ersetzen wir die Terminalsymbole a, b und c durch die Variablen A, B und C (außer wenn sie alleine auf der rechten Seite einer Regel vorkommen) und fügen die Regeln $A \to a$, $B \to b$, $C \to c$ hinzu:

$$S \to c, ABC, AY, BX, CS;\ \ X \to a, AS, BXX;$$
$$Y \to b, BS, AYY;\ \ A \to a;\ \ B \to b;\ \ C \to c.$$

Ersetze nun die Regeln $S \to ABC$, $X \to BXX$ und $Y \to AYY$ durch die Regeln $S \to AS'$, $S' \to BC$, $X \to BX'$, $X' \to XX$ und $Y \to AY'$, $Y' \to YY$:

$$S \to c, AS', AY, BX, CS;\ \ S' \to BC;$$
$$X \to a, AS, BX';\ \ X' \to XX;\ \ Y \to b, BS, AY';\ \ Y' \to YY;$$
$$A \to a;\ B \to b;\ C \to c.$$

◁

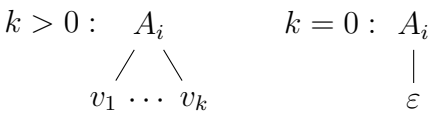
Für den Beweis des Pumping-Lemmas benötigen wir noch den Begriff des Syntaxbaums (auch **Ableitungsbaum** genannt, engl. *parse tree*).

Definition 84. *Wir ordnen einer Ableitung*

$$\underline{A_0} \Rightarrow l_1\underline{A_1}r_1 \Rightarrow \cdots \Rightarrow l_{m-1}\underline{A_{m-1}}r_{m-1} \Rightarrow \alpha_m.$$

den Syntaxbaum T_m zu, wobei die Bäume $T_0, \ldots, T_m$ induktiv wie folgt definiert sind:

- *T_0 besteht aus einem einzigen Knoten, der mit A_0 markiert ist.*
- *Wird im $(i+1)$-ten Ableitungsschritt die Regel $A_i \to v_1 \cdots v_k$ mit $v_j \in \Sigma \cup V$ für $j = 1, \ldots, k$ angewandt, so ensteht T_{i+1} aus T_i, indem wir das Blatt A_i in T_i durch folgenden Unterbaum ersetzen:*

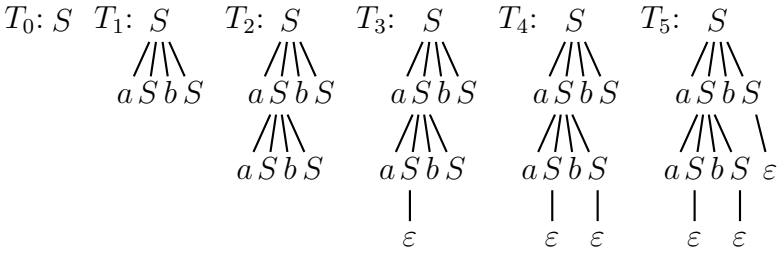


- *Hierbei stellen wir uns die Kanten von oben nach unten gerichtet und die Kinder $v_1 \cdots v_k$ von links nach rechts geordnet vor.*

Beispiel 85. *Betrachte die Grammatik $G = (\{S\}, \{a, b, c\}, \{S \to aSbS, \varepsilon\}, S)$ und die Ableitung*

$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aaSb\underline{S}bS \Rightarrow aa\underline{S}bbS \Rightarrow aabb\underline{S} \Rightarrow aabb.$$

Die zugehörigen Syntaxbäume sind dann



Die Satzform α_i ergibt sich aus T_i, indem wir die Blätter von T_i von links nach rechts zu einem Wort zusammensetzen. ◁

Es ist klar, dass Ableitungen, die sich nur in der Reihenfolge der Regelanwendungen unterscheiden, auf denselben Syntaxbaum führen. Dies bedeutet, dass aus einem Syntaxbaum die zugrunde liegende Ableitung nicht eindeutig rekonstruierbar ist. Dies ändert sich, wenn wir die Reihenfolge der Regelanwendungen festlegen.

Definition 86. *Sei $G = (V, \Sigma, P, S)$ eine kontextfreie Grammatik.*

a) Eine Ableitung

$$\alpha_0 = l_0\underline{A_0}r_0 \Rightarrow l_1\underline{A_1}r_1 \Rightarrow \cdots \Rightarrow l_{m-1}\underline{A_{m-1}}r_{m-1} \Rightarrow \alpha_m.$$

*heißt **Linksableitung** von α (kurz $\alpha_0 \Rightarrow_L^* \alpha_m$), falls in jedem Ableitungsschritt die am weitesten links stehende Variable ersetzt wird, d.h. es gilt $l_i \in \Sigma^*$ für $i = 0, \ldots, m-1$.*

*b) **Rechtsableitungen** $\alpha_0 \Rightarrow_R^* \alpha_m$ sind analog definiert.*

*c) G heißt **mehrdeutig**, wenn es ein Wort $x \in L(G)$ gibt, das zwei verschiedene Linksableitungen $S \Rightarrow_L^* x$ hat.*

Es ist leicht zu sehen, dass für alle Wörter $x \in \Sigma^*$ folgende Äquivalenzen gelten:

$$x \in L(G) \;\Leftrightarrow\; S \Rightarrow^* x \;\Leftrightarrow\; S \Rightarrow_L^* x \;\Leftrightarrow\; S \Rightarrow_R^* x.$$

Beispiel 87. *Wir betrachten nochmals die Grammatik $G = (\{S\}, \{a, b, c\}, \{S \to aSbS, \varepsilon\}, S)$ aus dem letzten Beispiel. Es gibt insgesamt zehn Ableitungen, die auf den dort abgebildeten Syntaxbaum T_5 führen:*

$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aaSb\underline{S}bS \Rightarrow aa\underline{S}bbS \Rightarrow aabb\underline{S} \Rightarrow aabb$$
$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aaSb\underline{S}bS \Rightarrow aaSbb\underline{S} \Rightarrow aa\underline{S}bb \Rightarrow aabb$$
$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aa\underline{S}bSbS \Rightarrow aabSbS \Rightarrow aabb\underline{S} \Rightarrow aabb$$
$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aa\underline{S}bSbS \Rightarrow aabSb\underline{S} \Rightarrow aab\underline{S}b \Rightarrow aabb$$
$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aaSbSb\underline{S} \Rightarrow aa\underline{S}bSb \Rightarrow aab\underline{S}b \Rightarrow aabb$$
$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aaSbSb\underline{S} \Rightarrow aaSb\underline{S}b \Rightarrow aa\underline{S}bb \Rightarrow aabb$$
$$\underline{S} \Rightarrow aSb\underline{S} \Rightarrow a\underline{S}b \Rightarrow aa\underline{S}bSb \Rightarrow aab\underline{S}b \Rightarrow aabb$$
$$\underline{S} \Rightarrow aSb\underline{S} \Rightarrow a\underline{S}b \Rightarrow aaSb\underline{S}b \Rightarrow aa\underline{S}bb \Rightarrow aabb.$$

Darunter sind genau eine Links- und genau eine Rechtsableitung, nämlich

$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aa\underline{S}bSbS \Rightarrow aab\underline{S}bS \Rightarrow aabb\underline{S} \Rightarrow aabb$$

und

$$\underline{S} \Rightarrow aSb\underline{S} \Rightarrow a\underline{S}b \Rightarrow aaSb\underline{S}b \Rightarrow aa\underline{S}bb \Rightarrow aabb.$$

Die Grammatik G ist eindeutig. Dies liegt daran, dass in keiner Satzform von G die Variable S von einem a gefolgt wird. Daher muss jede Linksableitung eines Wortes $x \in L(G)$ die am weitesten links stehende Variable der aktuellen Satzform $\alpha S \beta$ genau dann nach $aSbS$ expandieren, falls das Präfix α in x von einem a gefolgt wird.

Dagegen ist die Grammatik $G' = (\{S\}, \{a, b, c\}, \{S \to aSbS, ab, \varepsilon\}, S)$ mehrdeutig, da das Wort $x = ab$ zwei verschiedene Linksableitungen hat:

$$\underline{S} \Rightarrow ab \text{ und } \underline{S} \Rightarrow a\underline{S}bS \Rightarrow ab\underline{S} \Rightarrow ab.$$

◁

Bemerkung 88.

- *Aus einem Syntaxbaum ist die zugehörige Linksableitung eindeutig rekonstruierbar. Daher führen unterschiedliche Linksableitungen auch auf unterschiedliche Syntaxbäume. Linksableitungen und Syntaxbäume entsprechen sich also eineindeutig. Ebenso Rechtsableitungen und Syntaxbäume.*
- *Ist T Syntaxbaum einer CNF-Grammatik, so hat jeder Knoten in T höchstens zwei Kinder (d.h. T ist ein Binärbaum).*

3.2 Das Pumping-Lemma für kontextfreie Sprachen

In diesem Abschnitt beweisen wir das Pumping-Lemma für kontextfreie Sprachen. Dabei nützen wir die Tatsache aus, dass die Syntaxbäume einer CNF-Grammatik Binärbäume sind.

Definition 89. *Die **Tiefe** eines Baumes mit Wurzel w ist die maximale Pfadlänge von w zu einem Blatt.*

Lemma 90. *Ein Binärbaum B der Tiefe k hat höchstens 2^k Blätter.*

Beweis. Wir führen den Beweis durch Induktion über k.

$k = 0$: Ein Baum der Tiefe 0 kann nur einen Knoten haben.

$k \rightsquigarrow k+1$: Sei B ein Binärbaum der Tiefe $k+1$. Dann hängen an B's Wurzel maximal zwei Teilbäume. Da deren Tiefe $\leq k$ ist, haben sie nach IV höchstens 2^k Blätter. Also hat $B \leq 2^{k+1}$ Blätter. ■

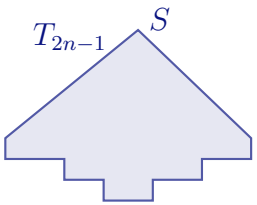
Korollar 91. *Ein Binärbaum B mit mehr als 2^{k-1} Blättern hat mindestens Tiefe k.*

Beweis. Würde B mehr als 2^{k-1} Blätter und eine Tiefe $\leq k-1$ besitzen, so würde dies im Widerspruch zu Lemma 90 stehen. ■

Satz 92 (Pumping-Lemma für kontextfreie Sprachen)**.**
Zu jeder kontextfreien Sprache L gibt es eine Zahl l, so dass sich alle Wörter $z \in L$ mit $|z| \geq l$ in $z = uvwxy$ zerlegen lassen mit

1. *$vx \neq \varepsilon$,*
2. *$|vwx| \leq l$ und*
3. *$uv^iwx^iy \in L$ für alle $i \geq 0$.*

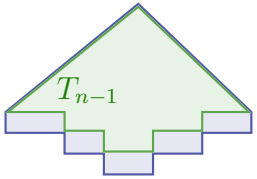
Beweis. Sei $G = (V, \Sigma, P, S)$ eine CNF-Grammatik für $L \setminus \{\varepsilon\}$. Dann existiert in G für jedes Wort $z = z_1 \cdots z_n \in L$ mit $n \geq 1$, eine Ableitung

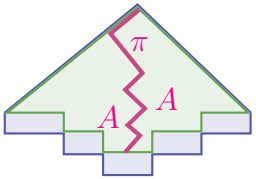


$$S = \alpha_0 \Rightarrow \alpha_1 \cdots \Rightarrow \alpha_m = z.$$

Da G in CNF ist, werden hierbei $n-1$ Regeln der Form $A \to BC$ und n Regeln der Form $A \to a$ angewandt, d.h. $m = 2n-1$ und z hat den Syntaxbaum T_{2n-1}.

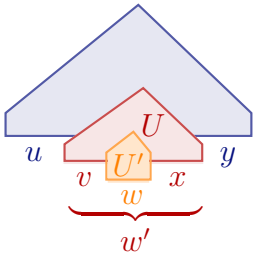
Wir können annehmen, dass zuerst alle Regeln der Form $A \to BC$ und danach die Regeln der Form $A \to a$ zur Anwendung kommen. Dann besteht die Satzform α_{n-1} aus n Variablen und der Syntaxbaum T_{n-1} hat ebenfalls n Blätter.



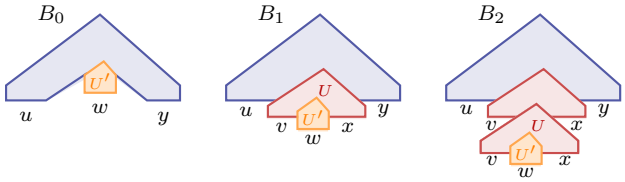


Setzen wir $l = 2^k$, wobei $k = \|V\|$ ist, so hat T_{n-1} im Fall $n \geq l$ mindestens $l = 2^k > 2^{k-1}$ Blätter und daher mindestens die Tiefe k. Sei π ein von der Wurzel ausgehender Pfad maximaler Länge in T_{n-1}. Dann hat π die Länge $\geq k$ und unter den letzten $k+1$ Knoten von π müssen zwei mit derselben Variablen A markiert sein.

Seien U und U' die von diesen Knoten ausgehenden Unterbäume des vollständigen Syntaxbaums T_{2n-1}. Nun zerlegen wir z wie folgt. w' ist das Teilwort von $z = uw'y$, das von U erzeugt wird und w ist das Teilwort von $w' = vwx$, das von U' erzeugt wird. Jetzt bleibt nur noch zu zeigen, dass diese Zerlegung die geforderten 3 Eigenschaften erfüllt.



- Da U mehr Blätter hat als U', ist $vx \neq \varepsilon$ (Bedingung 1).
- Da der Baum $U^* = U \cap T_{n-1}$ die Tiefe $\leq k$ hat (andernfalls wäre π nicht maximal), hat U^* höchstens $2^k = l$ Blätter. Da U^* genau $|vwx|$ Blätter hat, folgt $|vwx| \leq l$ (Bedingung 2).
- Für den Nachweis von Bedingung 3 lassen sich schließlich Syntaxbäume B^i für die Wörter uv^iwx^iy, $i \geq 0$, wie folgt konstruieren:



B^0 entsteht also aus $B^1 = T_{2n-1}$, indem wir U durch U' ersetzen, und B^{i+1} entsteht aus B^i, indem wir U' durch U ersetzen. ■

Beispiel 93. *Betrachte die Sprache $L = \{a^nb^n | n \geq 0\}$. Dann lässt sich jedes Wort $z = a^nb^n$ mit $|z| \geq 2$ pumpen: Zerlege $z = uvwxy$ mit $u = a^{n-1}$, $v = a$, $w = \varepsilon$, $x = b$ und $y = b^{n-1}$.* ◁

Beispiel 94. *Die Sprache $\{a^nb^nc^n \mid n \geq 0\}$ ist nicht kontextfrei. Für eine vorgegebene Zahl $l \geq 0$ hat nämlich $z = a^lb^lc^l$ die Länge $|z| = 3l \geq l$. Dieses Wort lässt sich aber nicht pumpen, da für jede Zerlegung $z = uvwxy$ mit $vx \neq \varepsilon$ und $|vwx| \leq l$ das Wort $z' = uv^2wx^2y$ nicht zu L gehört:*

- *Wegen $vx \neq \varepsilon$ ist $|z| < |z'|$.*
- *Wegen $|vwx| \leq l$ kann in vx nicht jedes der drei Zeichen a, b, c vorkommen.*
- *Kommt aber in vx beispielsweise kein a vor, so ist*

$$\#_a(z') = \#_a(z) = l = |z|/3 < |z'|/3,$$

also kann z' nicht zu L gehören. ◁

Satz 95. *Die Klasse* CFL *ist nicht abgeschlossen unter Durchschnitt und Komplement.*

Beweis. Die beiden Sprachen

$$L_1 = \{a^nb^mc^m \mid n, m \geq 0\} \quad \text{und} \quad L_2 = \{a^nb^nc^m \mid n, m \geq 0\}$$

sind kontextfrei. Nicht jedoch $L_1 \cap L_2 = \{a^nb^nc^n \mid n \geq 0\}$. Also ist CFL nicht unter Durchschnitt abgeschlossen.

Da CFL zwar unter Vereinigung aber nicht unter Schnitt abgeschlossen ist, kann CFL wegen de Morgan nicht unter Komplementbildung abgeschlossen sein. ■

3.3 Der CYK-Algorithmus

In diesem Abschnitt stellen wir einen effizienten Algorithmus zur Lösung des Wortproblems für kontextfreie Grammatiken vor, das wie folgt definiert ist.

Wortproblem für kontextfreie Grammatiken:

Gegeben: Eine kontextfreie Grammatik G und ein Wort x.

Gefragt: Ist $x \in L(G)$?

Wir lösen das Wortproblem, indem wir G zunächst in Chomsky-Normalform bringen und dann den nach seinen Autoren Cocke, Younger und Kasami benannten CYK-Algorithmus anwenden, welcher auf dem Prinzip der Dynamischen Programmierung beruht.

Satz 96. *Das Wortproblem für kontextfreie Grammatiken ist effizient entscheidbar.*

Beweis. Seien eine Grammatik $G = (V, \Sigma, P, S)$ und ein Wort $x = x_1 \cdots x_n$ gegeben. Falls $x = \varepsilon$ ist, können wir effizient prüfen, ob $S \Rightarrow^* \varepsilon$ gilt. Andernfalls transformieren wir G in eine CNF-Grammatik G' für die Sprache $L(G) \setminus \{\varepsilon\}$. Chomsky-Normalform. Es lässt sich leicht verifizieren, dass die nötigen Umformungsschritte effizient ausführbar sind. Nun setzen wir den CYK-Algorithmus auf das Paar (G', x) an, der die Zugehörigkeit von x zu $L(G')$ wie folgt entscheidet.
Bestimme für $l = 1, \ldots, n$ und $k = 1, \ldots, n - l + 1$ die Menge

$$V_{l,k}(x) = \{A \in V \mid A \Rightarrow^* x_k \cdots x_{k+l-1}\}$$

aller Variablen, aus denen das mit x_k beginnende Teilwort $x_k \cdots x_{k+l-1}$ von x der Länge l ableitbar ist. Dann gilt offensichtlich $x \in L(G') \Leftrightarrow S \in V_{n,1}(x)$.
Für $l = 1$ ist

$$V_{1,k}(x) = \{A \in V \mid A \to x_k\}$$

und für $l = 2, \cdots, n$ ist

$$V_{l,k}(x) = \{A \in V \mid \exists l' < l \, \exists B \in V_{l',k}(x) \, \exists C \in V_{l-l',k+l'}(x) \colon A \to BC\}.$$

Eine Variable A gehört also genau dann zu $V_{l,k}(x)$, $l \geq 2$, falls eine Zahl $l' \in \{1, \ldots, l-1\}$ und eine Regel $A \to BC$ existieren, so dass $B \in V_{l',k}(x)$ und $C \in V_{l-l',k+l'}(x)$ sind.
Da der Zeitaufwand für die Berechnung der Menge $V_{l,k}(x)$ durch $O(l|G'|)$ beschränkt ist, und insgesamt $n(n+1)/2$ solche Mengen zu bestimmen sind, lässt sich die Zeitkomplexität durch $O(n^3|G'|)$ abschätzen ■

Algorithmus `CYK`(G, x)

```
1  Input: CNF-Grammatik G = (V, Σ, P, S) und ein Wort
       x = x_1 ··· x_n
2    for k := 1 to n do
3      V_{1,k} := {A ∈ V | A → x_k ∈ P}
4    for l := 2 to n do
5      for k := 1 to n − l + 1 do
6        V_{l,k} := ∅
7        for l' := 1 to l − 1 do
8          for all A → BC ∈ P do
9            if B ∈ V_{l',k} and C ∈ V_{l−l',k+l'} then
10             V_{l,k} := V_{l,k} ∪ {A}
11   if S ∈ V_{n,1} then accept else reject
```

Der CYK-Algorithmus lässt sich leicht dahingehend modifizieren, dass er im Fall $x \in L(G)$ auch einen Syntaxbaum T von x ausgibt. Hierzu genügt es, zu jeder Variablen A in $V_{l,k}$ den Wert von l' und die Regel $A \to BC$ zu speichern, die zur Aufnahme von A in $V_{l,k}$ geführt haben. Im Fall $S \in V_{n,1}(x)$ lässt sich dann mithilfe dieser Information leicht ein Syntaxbaum T von x konstruieren.

Beispiel 97. *Betrachte die CNF-Grammatik mit den Produktionen*

$S \to AS', AY, BX, CS, c$; $S' \to BC$; $X \to AS, BX', a$; $X' \to XX$; $Y \to BS, AY', b$; $Y' \to YY$; $A \to a$; $B \to b$; $C \to c$.

Dann erhalten wir für das Wort $x = abb$ folgende Mengen $V_{l,k}$:

x_k:	a	b	b
l:1	$\{X, A\}$	$\{Y, B\}$	$\{Y, B\}$
2	$\{S\}$	$\{Y'\}$	
3	$\{Y\}$		

Wegen $S \notin V_{3,1}(abb)$ ist $x \notin L(G)$. Dagegen gehört das Wort $y = aababb$ wegen $S \in V_{6,1}(aababb)$ zu $L(G)$:

a	a	b	a	b	b
$\{X, A\}$	$\{X, A\}$	$\{Y, B\}$	$\{X, A\}$	$\{Y, B\}$	$\{Y, B\}$
$\{X'\}$	$\{S\}$	$\{S\}$	$\{S\}$	$\{Y'\}$	
$\{X\}$	$\{X\}$	$\{Y\}$	$\{Y\}$		
$\{X'\}$	$\{S\}$	$\{Y'\}$			
$\{X\}$	$\{Y\}$				
$\{S\}$					

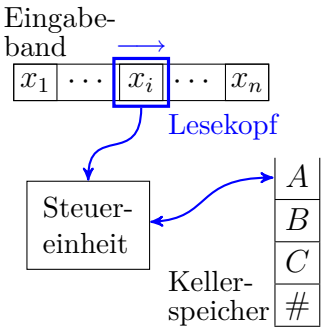
◁

3.4 Kellerautomaten

In diesem Abschnitt befassen wir uns mit der Frage, wie sich das Maschinenmodell des DFA erweitern lässt, um die Sprache $L = \{a^n b^n \mid n \geq 0\}$ und alle anderen kontextfreien Sprachen erkennen zu können. Dass ein DFA die Sprache $L = \{a^n b^n \mid n \geq 0\}$ nicht erkennen kann, liegt an seinem beschränkten Speichervermögen, das zwar von L aber nicht von der Eingabe abhängen darf.

Um L erkennen zu können, reicht bereits ein so genannter Kellerspeicher (Stapel, engl. *stack, pushdown memory*) aus. Dieser erlaubt nur den Zugriff auf die höchste belegte Speicheradresse. Ein Kellerautomat

- verfügt über einen Kellerspeicher,
- kann ε-Übergänge machen,
- liest in jedem Schritt das aktuelle Eingabezeichen und das oberste Kellersymbol,
- kann das oberste Kellersymbol entfernen (durch eine **pop-Operation**) und
- danach beliebig viele Symbole einkellern (mittels **push-Operationen**).



Für eine Menge M bezeichne $\mathcal{P}_e(M)$ die Menge aller endlichen Teilmengen von M, d.h.

$$\mathcal{P}_e(M) = \{A \subseteq M \mid A \text{ ist endlich}\}.$$

Definition 98. *Ein **Kellerautomat** (kurz: PDA; pushdown automaton) wird durch ein 6-Tupel $M = (Z, \Sigma, \Gamma, \delta, q_0, \#)$ beschrieben, wobei*

- *$Z \neq \emptyset$ eine endliche Menge von **Zuständen**,*
- *Σ das **Eingabealphabet**,*
- *Γ das **Kelleralphabet**,*
- *$\delta : Z \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to \mathcal{P}_e(Z \times \Gamma^*)$ die **Überführungsfunktion**,*
- *$q_0 \in Z$ der **Startzustand** und*
- *$\# \in \Gamma$ das **Kelleranfangszeichen** ist.*

Wenn q der momentane Zustand, A das oberste Kellerzeichen und $u \in \Sigma$ das nächste Eingabezeichen (bzw. $u = \varepsilon$) ist, so kann M im Fall $(p, B_1 \cdots B_k) \in \delta(q, u, A)$

- in den Zustand p wechseln,

- den Lesekopf auf dem Eingabeband um $|u|$ Positionen vorrücken und
- das Zeichen A im Keller durch die Zeichenfolge $B_1 \cdots B_k$ ersetzen.

Hierfür sagen wir auch, M führt die **Anweisung** $quA \to pB_1 \cdots B_k$ aus. Da im Fall $u = \varepsilon$ kein Eingabezeichen gelesen wird, spricht man auch von einem **spontanen** Übergang (oder **ε-Übergang**). Eine **Konfiguration** wird durch ein Tripel

$$K = (q, x_i \cdots x_n, A_1 \cdots A_l) \in Z \times \Sigma^* \times \Gamma^*$$

beschrieben und besagt, dass

- q der momentane Zustand,
- $x_i \cdots x_n$ der ungelesene Rest der Eingabe und
- $A_1 \cdots A_l$ der aktuelle Kellerinhalt ist (A_1 steht oben).

Eine Anweisung $quA_1 \to pB_1 \cdots B_k$ (mit $u \in \{\varepsilon, x_i\}$) überführt die Konfiguration K in die **Folgekonfiguration**

$$K' = (p, x_j \cdots x_n, B_1 \cdots B_k A_2 \cdots A_l) \text{ mit } j = i + |u|.$$

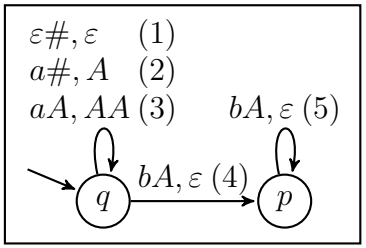
Hierfür schreiben wir auch kurz $K \vdash K'$. Die reflexive, transitive Hülle von $\vdash$ bezeichnen wir wie üblich mit $\vdash^*$. Die von M **akzeptierte** oder **erkannte Sprache** ist

$$L(M) \;=\; \{x \in \Sigma^* \mid \exists p \in Z : (q_0, x, \#) \vdash^* (p, \varepsilon, \varepsilon)\}.$$

Ein Wort x wird also genau dann von M akzeptiert, wenn es eine Rechnung (Folge von Konfigurationen) von M bei Eingabe x gibt, die ausgehend von der **Startkonfiguration** $(q_0, x, \#)$ das gesamte Wort bis zum Ende liest und den Keller leert. Man beachte, dass bei leerem Keller kein weiterer Übergang mehr möglich ist.

Beispiel 99. *Sei $M = (Z, \Sigma, \Gamma, \delta, q, \#)$ ein PDA mit $Z = \{q, p\}$, $\Sigma = \{a, b\}$, $\Gamma = \{A, \#\}$ und den Anweisungen*

$$\begin{array}{llll} \delta : q\varepsilon\# \to q & (1) & qa\# \to qA & (2) \\ qaA \to qAA & (3) & qbA \to p & (4) \\ pbA \to p & (5) & & \end{array}$$



Dann akzeptiert M die Eingabe aabb:

$$(q, aabb, \#) \underset{(2)}{\vdash} (q, abb, A) \underset{(3)}{\vdash} (q, bb, AA) \underset{(4)}{\vdash} (p, b, A) \underset{(5)}{\vdash} (p, \varepsilon, \varepsilon).$$

Allgemein akzeptiert M das Wort $x = a^n b^n$ mit folgender Rechnung:

$n = 0$: $(q, \varepsilon, \#) \underset{(1)}{\vdash} (p, \varepsilon, \varepsilon)$.

$n \geq 1$: $(q, a^n b^n, \#) \underset{(2)}{\vdash} (q, a^{n-1} b^n, A) \overset{n-1}{\underset{(3)}{\vdash}} (q, b^n, A^n) \underset{(4)}{\vdash} (p, b^{n-1}, A^{n-1}) \overset{n-1}{\underset{(5)}{\vdash}} (p, \varepsilon, \varepsilon)$.

Dies zeigt $\{a^n b^n \mid n \geq 0\} \subseteq L(M)$. Als nächstes zeigen wir, dass jede von M akzeptierte Eingabe $x = x_1 \ldots x_n$ die Form $x = a^m b^m$ hat.

Ausgehend von der Startkonfiguration $(q, x, \#)$ sind nur die Anweisungen (1) oder (2) möglich. Da Anweisung (1) den Keller leert, ohne ein Zeichen zu lesen, muss x in diesem Fall gleich $\varepsilon = a^0 b^0$ sein.

Falls M mit Anweisung (2) beginnt, muss M mittels Anweisung (4) in den Zustand p gelangen, da andernfalls der Keller nicht geleert wird. Dies geschieht, sobald M nach Lesen von $m \geq 1$ a's das erste b liest:

$$(q, x_1 \ldots x_n, \#) \underset{(2)}{\vdash} (q, x_2 \ldots x_n, A) \overset{m-1}{\underset{(3)}{\vdash}} (q, x_{m+1} \cdots x_n, A^m) \underset{(4)}{\vdash} (p, x_{m+2} \cdots x_n, A^{m-1})$$

mit $x_1 = x_2 = \cdots = x_m = a$ und $x_{m+1} = b$. Um den Keller zu leeren, muss M nun noch genau $m - 1$ b's lesen, weshalb x auch in diesem Fall die Form $a^m b^m$ hat. ◁